Adrià Sánchez Moreno

Grau en Enginyeria en Tecnologies Industrials

Mes Any

Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Desenvolupament de videojocs 2D amb Unity

MEMÒRIA

Autor: Adrià Sánchez Moreno
Director/s: Lluis Solano Albajes

Convocatòria: Gener 2016

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte consisteix en el disseny d'un videojoc per computador, així com la seva implementació utilitzant el software la plataforma de desenvolupament Unity, realitzat amb el llenguatge de programació C#. El resultat és un prototip d'aquest videojoc disponible per a sistemes operatius Windows i Linux.

Es tracta d'un videojoc per un jugador, del tipus plataformes i acció en vista lateral 2D, basat en mapes de bits anomenats *sprites*. En aquesta memòria s'explica amb detall les característiques que es desitgen en el joc final, així com la metodologia emprada per implementar algunes d'aquestes característiques en el prototip.

En la realització del prototip, s'ha utilitzat principalment *Unity* per implementar tots els elements que el conformen, excepte en la creació de nivells. Per realitzar aquesta tasca s'ha utilitzat el software *Tiled* per crear mapes basats en *tiles*, que són mapes de bits d'una mateixa grandària que es combinen en forma de graella per crear el mapa de bits que representa el nivell complert. Posteriorment, aquests nivells s'han importat a *Unity* mitjançant la utilitat *Tiled2Unity*.

Sumari

1. Glossari

- ***Sprite:*** Un *sprite* és, als videojocs, un element gràfic que es pot desplaçar sobre la pantalla. En principi, un *sprite* és parcialment transparent, i pot ser animat (és a dir, és format de diversos mapes de bits que apareixen uns sobre els altres). El fons de la pantalla constitueix generalment l'escenari i els *sprites* són els personatges i objectes que se superposen al fons de la pantalla i que es desplacen. Un *sprite* també pot de vegades passar darrere un element del fons de la pantalla.

- ***Tile:*** Els *tiles* són mapes de bits de forma quadrada, rectangular o hexagonal utilitzats per representar l'àrea de joc en alguns videojocs. El conjunt complert dels *tiles* disponibles per una àrea concreta s'anomena *tileset.* Els *tiles* es disposen de manera adjacent entre si en una graella i, usualment, alguns *tiles* es poden superposar als altres, per exemple quan un *tile* que representa el terreny es superposa a la imatge de fons.

- ***Script:*** En informàtica, un script o guió és un programa, normalment simple, que generalment s'emmagatzema en un arxiu de text pla. Els scripts són gairebé sempre interpretats, però no tot programa interpretat és considerat un script. L'ús habitual dels scripts és realitzar tasques com combinar components, interactuar amb el sistema operatiu o amb l'usuari.

- ***Mecàniques de joc:*** Les mecàniques de joc es defineixen com les regles o conjunt de regles que tenen com a objectiu aconseguir una sèrie de resultats coherents dins del propi joc. Els jocs complexes, com els jocs de rol, poden disposar d'una gran quantitat de regles interconnectades entre si.

2. Prefaci

2.1. Origen i motivació del projecte

La principal motivació per aquest projecte és la meva gran afició pel món dels videojocs, així com la bona experiència que han suposat les assignatures relacionades amb la programació que he cursat a l'escola. Les assignatures obligatòries de primer i segon del *Grau en Enginyeria en Tecnologies Industrials, Fonaments d'informàtica* i *Informàtica* van despertar en mi un gran interès per la programació, i durant la carrera he anat cursant diferents assignatures optatives i projectes relacionats amb aquest camp.

No és la meva primera experiència en el camp del desenvolupament de videojocs, doncs una de les assignatures optatives que he cursat és *Jocs per Computador*, on vam crear un petit joc basat en *Flash*, amb el llenguatge *ActionScript*. En ser una assignatura optativa, el nombre d'hores de dedicació és bastant limitat, i vaig pensar que aquest treball seria una gran oportunitat per aprofundir en el tema.

Un altre motiu pel que he decidit realitzar aquest projecte és diversificar els meus coneixements en quant a llenguatges de programació, doncs en la majoria d'assignatures del grau hem utilitzat Python.

3. Introducció

En els últim anys, els videojocs s'estan consolidant a nivell mundial com la principal industria d'oci digital i interactiu, amb una quota de mercat molt superior a les del cinema i la música.

Malgrat la situació econòmica actual, el sector dels videojocs ha mantingut la seva aposta per la innovació, adaptant-se a les noves tendències de consum i trobant nous segments de mercat per expandir el creixement del negoci. Segons l'*Associació Espanyola de Videojocs* (*AEVI*), Espanya es troba en cinquena posició pel que fa a ingressos obtinguts en la Unió Europea i en desena posició a nivell mundial, amb uns ingressos totals de 996 milions d'euros l'any 2014 **[1]**.

Actualment existeix una gran varietat de jocs, per a tot tipus de dispositius electrònics i per tots els gustos, així com una gran quantitat d'eines que faciliten el seu desenvolupament. Una d'aquestes eines és *Unity*, que a més de incorporar nombroses utilitats per la realització de videojocs, permet que un mateix projecte pugui ser exportat a una gran multitud de plataformes de manera directa. Òbviament existeixen altres alternatives en el desenvolupament de videojocs que també s'han de considerar, en la memòria s'explica els motius per utilitzar *Unity*.

3.1. Objectius del projecte

L'objectiu principal del projecte és descriure la metodologia a seguir per desenvolupar un videojoc 2D, des del seu disseny conceptual fins que s'obté el software final que és consumit pel jugador.

Un altre objectiu important, relacionat amb la implementació del videojoc, és detallar les facilitats que aporta *Unity* a l'hora de traslladar les idees definides en la fase de disseny a la programació del videojoc, així com els inconvenients que es presenten.

3.2. Abast del projecte

L'abast del projecte arriba fins el desenvolupament d'un primer prototip del que seria el videojoc final, incorporant algunes de les característiques principals del joc ideades a la fase de disseny. Un cop finalitzat el projecte, s'obtindrà aquest prototip per a les plataformes oportunes. Al projecte es descriuran quines han estat les eines utilitzades durant tot aquest procés, així com els recursos que s'han utilitzat durant el desenvolupament.

4. Disseny conceptual del videojoc

La primera etapa en el desenvolupament de videojocs és el disseny conceptual del joc. En aquesta fase es defineixen totes les característiques que contindrà el producte final, com per exemple l'argument de la trama, les mecàniques jugables que s'incorporaran o els personatges que apareixen.

4.1. Descripció general

En primer lloc cal definir els aspectes més generals del joc com l'argument, el gènere al que pertanyerà, els jocs que es prenen com a referència, etc. A continuació es mostra la descripció del cas d'estudi.

Títol del joc: Exam Rush

Gènere: Plataformes i acció; Rol

Plataformes: Windows, Mac OS, Linux

Influències: Megaman, Castlevania, Shovel Knight.

Aspecte visual: Joc 2D amb desplaçament lateral, basat en *sprites*.

Argument: Un estudiant està preparant els exàmens finals i es troba tant cansat que es queda adormit mentre estudia. Quan desperta es troba a un món on hi ha altres estudiants que han quedat adormits com ell, però no poden sortir ja que el mag Grimgrorn els manté atrapats. Un altre estudiant que ha estat atrapat durant molt temps, anomenat Toke, en veure el poder del protagonista, decideix cedir-li la seva millor creació, un vestit robòtic que el permetrà derrotar els monstres creats per en Grimgrorn per que els estudiants no puguin escapar del món dels somnis. La missió del protagonista és vèncer aquest enemic i aconseguir sortir d'aquest món abans de l'examen.

4.2. Mecàniques de joc

Mitjançant les mecàniques de joc es defineixen les condicions que s'han de donar per a que el jugador guanyi o perdi i per completar el joc. En els següents apartats es descriuen en detall les mecàniques que incorporarà el joc.

4.2.1. Condicions d'èxit i fallida

El joc es divideix en nivells. Cada nivell conté una sèrie de punts de control, on l'últim d'aquests punts es troba sempre just abans de la batalla amb l'enemic final del nivell. Si el jugador el derrota, es desbloqueja un nou nivell, finalitzant el joc quan l'usuari derroti a l'enemic final de l'últim nivell.

Durant el transcurs dels nivells el jugador ha d'eliminar o esquivar els diferents enemics que es troben en ell. Per a tal efecte disposa de projectils i diferents habilitats que es poden desbloquejar en el transcurs del joc. Al vèncer els enemics existeix la possibilitat d'aconseguir monedes, que serveixen per comprar diferents objectes a la botiga per millorar o personalitzar el personatge.

El jugador compta amb una certa quantitat de vida, i mor si aquesta arriba a zero. Si un enemic o un projectil llançat per algun d'ells toca al personatge, aquest perd vida en funció del poder de l'atac que l'ha ferit. Per curar-se, el jugador disposa d'un nombre limitat de pocions, que es poden comprar abans de jugar el nivell i que, addicionalment, es regeneren de forma limitada a cada punt de control dins dels nivells.

Si el personatge mor, perd totes les monedes que no havia gastat i ha de tornar al punt on va morir per recuperar-les. D'aquesta forma es penalitza la mort amb la possibilitat de perdre recursos valuosos per la progressió en el joc, incentivant als jugadors a millorar les seves habilitats.

D'igual manera també existeix la possibilitat de destruir els punts de control per aconseguir monedes extra en vèncer als enemics. Si un punt de control és destruït, queda anul·lat fins que el jugador completi el nivell o en surti d'ell mitjançant el menú. Amb aquesta mecànica es pretén que els jugadors que vulguin jugar d'una manera més arriscada es vegin recompensats.

Els nivells prèviament completats es podran tornar a jugar mitjançant un mapa del món que mostra tots els nivells, tot i que l'enemic final només apareixerà el primer cop que es juga. D'aquesta manera sempre es poden aconseguir els recursos necessaris per comprar els diferents objectes oferts a la botiga.

4.2.2. Millora progressiva del personatge

Les característiques del personatge van millorant durant el transcurs del joc, per adaptar-se a l'increment de dificultat que s'introdueix progressivament a mesura que es desbloquegen nous nivells. Aquestes millores es poden aconseguir de dues formes, recollint objectes de millora dins dels nivells o gastant les monedes obtingudes a la botiga.

Les característiques que es poden millorar són la quantitat màxima de vida, nivell d'escuts i munició d'armes secundàries. Addicionalment, també es poden adquirir noves habilitats i armes secundàries més poderoses.

4.2.3. Finals

Tal com es reflexa en l'argument del joc, l'objectiu del protagonista és sortir del món en que ha estat atrapat abans de l'hora del examen. Per aquest motiu s'introdueixen tres finals en funció de la manera en que el jugador ha superat el joc, basant-se principalment en el temps de compleció dels nivells. Per no penalitzar als jugadors que vulguin jugar de forma relaxada, els finals obtinguts depenen de la suma del millor temps en cada un dels nivells, que es poden repetir de manera indefinida. En tots els finals el protagonista aconsegueix alliberar a tots els estudiants atrapats al món dels somnis.

- **Final bàsic:** Aquest final es dóna quan el jugador supera el joc en un temps més gran que el temps límit. El protagonista no arriba a temps al examen i suspèn l'assignatura.

- **Final especial:** Les condicions per aconseguir aquest final és superar els nivells en un temps inferior al temps límit. El protagonista arriba a temps al examen, però ha estat tan ocupat lluitant per sortir del mon dels somnis que el suspèn.

- **Final definitiu:** Per aconseguir aquest final, a més de superar els nivells en un temps inferior al temps límit, cal recollir totes les millores i apunts disponibles al joc. El protagonista arriba a temps al examen, i degut a que ha adquirit molt coneixement durant la seva aventura, aconsegueix aprovar l'examen amb bona nota.

4.3. Personatges jugables

Els personatges són una part essencial dins d'un videojoc, doncs són els que permeten que es desenvolupi l'argument. Per tal d'aconseguir el seu objectiu, el protagonista interactua amb la resta de personatges, aliats o enemics, avançant així en la trama.

En el joc només hi ha com a personatge jugable l'estudiant, que és el protagonista de la història. Aquest disposa de diferents habilitats i armes que el permeten avançar pels nivells del joc.

4.3.1. Moviments

El jugador pot utilitzar els diferents moviments de que disposa el protagonista per progressar

a través dels escenaris. Els moviments disponibles són:

- **Córrer:** El personatge pot córrer sempre que estigui situat sobre una de les plataformes que conformen els nivells. Aquest és el moviment més habitual en el joc, doncs és necessari moure's de plataforma en plataforma per arribar al final dels nivells.

- **Saltar:** De manera similar a la capacitat de córrer, el jugador pot saltar sempre que estigui situat sobre una plataforma. Aquest moviment és utilitzat per arribar a una nova plataforma un cop s'arriba al final de l'anterior, així com per esquivar enemics o projectils llançats per aquests**.**

- **Rodar:** Es pot realitzar únicament quan el personatge està sobre una plataforma. El personatge roda sobre la seva espatlla i és invulnerable al dany durant el temps que es trobi en l'animació de rodar. Aquest moviment està pensat per evadir els atacs enemics o passar per sota de llocs estrets.

- **Ajupir-se:** Novament aquest és un moviment únicament realitzable quan el protagonista es troba sobre una plataforma. Aquest moviment s'utilitza bàsicament per esquivar atacs a distància, ja que el jugador no es pot moure mentre està ajupit.

4.3.2. Armes

A més d'utilitzar els moviments del personatge, el jugador té la possibilitat de matar als enemics per aconseguir monedes que el permeten obtenir diferents objectes i millores a la botiga. A tal efecte, el personatge disposa d'armes principals i secundàries, essent les principals d'us il·limitat i poder moderat i les secundàries d'us limitat i gran poder.

Les armes principals són el canó làser acoblat al vestit i la combinació de cops de puny i puntades de peu:

- **Canó làser:** El canó làser permet disparar projectils que causen dany als enemics al impactar amb ells. Els projectils viatgen en línia recta, en tots dos sentits si es disparen en horitzontal i únicament cap amunt en vertical. Aquesta arma conforma la principal arma a distància del personatge. El canó làser té dos modes d'utilització, el foc ràpid i el foc carregat.

    - **Foc ràpid:** Els projectils són disparats de manera continuada si el jugador manté pressionat el botó de foc ràpid. El dany provocat per aquest mode és inferior al del foc carregat per compensar la cadència de foc.

- **Foc carregat:** Quan es pressiona el botó de foc carregat, el personatge inicia una animació per preparar el dispar. Una vegada l'animació arriba al punt final, el làser està carregat i es pot disparar el projectil.

- **Cops físics:** El jugador pot realitzar petites combinacions de 3 cops físics, que causen danys als enemics si el jugador contacta físicament amb aquests. Durant la realització de la combinació el jugador és immune al dany per col·lisió, però pot perdre vida si és impactat per projectils enemics o objectes que provoquin dany.

Les armes secundàries són armes d'us limitat, per aquest motiu s'utilitza un sistema de munició. El nombre d'usos de cada arma ve determinat pel poder que té l'arma, i es poden recarregar els usos a la botiga o recollint la munició que es pot trobar al llarg dels nivells. Durant el transcurs del joc es poden aconseguir millores per augmentar la capacitat màxima de les armes secundàries. Les armes secundàries disponibles són:

- **Bomba:** El jugador la llança i després d'un temps determinat, explota causant dany en àrea amb un determinat radi al voltant de la posició de la bomba.

- **Làser:** Raig làser amb gran gruix que ocupa tot el llarg de la pantalla i afecta a tots els enemics que entrin amb contacte amb ell. Els usos són molt limitats degut al gran dany que provoca.

4.3.3. Habilitats

Addicionalment als moviments i les armes, el jugador també disposa d'habilitats especials que es desbloquegen a la botiga o mitjançant objectes amagats a l'escenari. A continuació es mostra una llista de les habilitats disponibles.

- **Doble salt:** Si el jugador pressiona el botó de salt un cop el personatge està a l'aire, podrà realitzar un altre salt que el permetrà arribar a plataformes més elevades.

- **Escut:** Quan s'utilitza aquesta habilitat, el protagonista és immune al dany durant 5 segons. Cada cop que es fa ús del escut, la quantitat d'energia del personatge disminueix, i només es pot recuperar aquesta energia descansant a la posada o mitjançant alguns objectes consumibles.

4.4. Personatges no jugables

Degut al caràcter del projecte, que té com a objectiu realitzar un prototip del joc i per tant l'art utilitzat serà d'us lliure, la majoria de personatges no jugables no estan totalment definits en

quant al seu aspecte. Tot i això, sí es determina tipus de personatges no jugables que s'implementaran al joc i les seves funcions.

Segons la interacció amb el jugador, existeixen dos tipus de personatges no jugables:

- **Aliats / neutrals:** En aquesta classificació s'inclouen els personatges que, o bé ajuden al protagonista a aconseguir el seu objectiu o bé no interfereixen en la compleció d'aquest. Un exemple d'aquest tipus de personatges seria Toke, que proporciona un vestit robòtic al protagonista o el propietari de la botiga on el jugador pot comprar pocions curatives o munició. Generalment aquest tipus de personatges es troben fora de les seccions jugables.

- **Enemics:** Aquest tipus de personatges s'oposen al protagonista i intenten que fracasi en la seva missió, i habitualment es troben a les parts on el jugador te control del personatge. L'enemic principal és Grimgrorn, que dóna ordres als altres enemics del joc. A la vegada, aquest tipus de personatges es poden classificar en dos categories, enemics comuns o enemics finals. Els enemics comuns, com el seu nom indica, apareixen de forma repetida al llarg dels nivells, mentre que els enemics finals juguen un rol a la trama del joc i apareixen únicament una vegada, generalment al final d'una fase determinada.

4.4.1. Aliats i personatges neutrals

Els personatges classificats dins d'aquesta categoria són aquells que no mostren un comportament hostil envers el protagonista. En base a la seva funció en la trama, es pot diferenciar entre aliats i personatges neutrals.

Els aliats són aquells personatges que intenten ajudar al personatge a aconseguir el seu objectiu, ja sigui per motius propis o per conveniència. La llista d'aliats és la següent:

- **Toke:** És un altre estudiant en la mateixa situació que el protagonista. Quan el protagonista arriba al món dels somnis, Toke ja ha estat atrapat durant un llarg període de temps. Degut a que les seves capacitats físiques no són bones, Toke ha dedicat el seu temps dins del món dels somnis a crear un vestit robòtic per quan aparegui un estudiant amb les qualitats adequades.

- **Waldan:** Waldan és l'estudiant que més temps ha estat atrapat dins del somni. Les seves capacitats físiques eren excepcionals, però degut a que ha persistit durant tant de temps al món dels somnis, ha envellit i no pot utilitzar el vestit robòtic de Toke de manera eficaç. Aquest personatge actua com a mestre, ensenyant al protagonista noves habilitats.

En quant als personatges neutrals, es tracta bàsicament de comerciants que tot i no sentir-se particularment identificats amb els objectius del jugador, no tenen motius per oposar-se al protagonista. Els personatges neutrals del joc són el propietari de la botiga i el propietari de la posada.

4.4.2. Enemics

Com s'ha esmentat amb anterioritat, els enemics es divideixen en enemics comuns i enemics finals. Els enemics comuns presents al joc són, segons la seva funció:

- **Patrulla:** El personatge camina contínuament, voltejant quan arriba a un punt determinat. Aquest comportament és utilitzat habitualment en combinació amb altres accions.

- **Patrulla amb dispar:** Realitza la mateixa funció que un enemic tipus *Patrulla*, però quan el personatge es troba al seu rang de visió dispara en la direcció d'aquest.

- **Patrulla amb envestida:** Novament, aquest enemic realitza patrulla fins que veu al personatge. Un cop ho fa, incrementa la seva velocitat i avança cap al personatge per causar-li danys per contacte.

- **Patrulla amb explosió:** Aquest tipus d'enemic és molt similar a un de tipus *Patrulla amb envestida*, però en lloc de buscar el dany per contacte, un cop està suficientment proper al personatge, s'immola provocant dany en un radi determinat. Habitualment són enemics voladors.

- **Fix amb dispar:** L'enemic no té capacitat de moviment i dispara projectils de forma continuada, obligant al jugador a esquivar-los.

Pel que fa als enemics finals, cada nivell excepte el tutorial compta amb un enemic final, amb un comportament i aspecte visual per definir. L'únic enemic final definit és Grimgrorn, que serà l'últim enemic a superar abans de completar el joc.

4.5. Objectes

Durant el transcurs del joc, s'utilitzen objectes que interactuen de diferents maneres amb el jugador. Aquests objectes es poden classificar segons la seva funció de la següent manera:

- **Objectes col·leccionables:** Si el personatge col·lideix amb un objecte col·leccionable, aquest desapareix i s'afegeix al nombre d'objectes que el jugador posseeix. Un exemple

d'objecte col·leccionable serien les monedes.

- **Objectes consumibles:** Els objectes d'aquest tipus desapareixen un cop utilitzats, i habitualment creen algun efecte sobre el jugador o el seu entorn. Els objectes consumibles poden ser emmagatzemats per el jugador per així gaudir del seu efecte en varies ocasions, com per exemple les pocions de vida.

- **Objectes interactius:** El personatge pot interactuar amb aquests objectes si entra en contacte amb ells. La interacció que es duu a terme depèn de l'objecte en qüestió, per exemple si és una escala el jugador pot pujar o baixar per ella.

4.5.1. Objectes col·leccionables

Com s'ha mencionat amb anterioritat els objectes col·leccionables són acumulats pel personatge principal, complint diferents funcions en cada cas. En general els objectes d'aquest tipus són presents per defecte als nivells tot i que en alguns casos poden ser obtinguts quan es derrota a un enemic. Els tipus d'objectes col·leccionables presents en el joc són:

- **Monedes:** Les monedes permeten la compra de tot tipus d'objectes beneficiosos per al jugador, com per exemple pocions per recuperar salut o munició per les armes secundàries.

- **Munició per armes secundàries:** Les armes secundàries són armes d'us limitat, per aquest motiu s'utilitza un sistema de munició. Quan es recull munició es restaura un nombre d'usos determinat, que varia depenent de l'arma.

- **Millores:** Com el seu nom indica, les millores permeten augmentar les capacitats del protagonista. Normalment les millores es troben amagades per tal que el jugador hagi d'explorar de manera exhaustiva el nivell que està jugant, tot i que algunes millores es poden comprar amb monedes. Els tipus de millores que es poden obtenir són:

    - **Millora de salut:** augmenta la salut màxima del personatge.

    - **Millora de munició:** augmenta la munició màxima de les armes secundàries. Cada tipus d'arma secundària té la seva pròpia millora.

    - **Millora de capacitat de pocions:** permet tenir més pocions regeneradores de salut.

    - **Millora de dany:** augmenta el dany de les armes principals. Cada arma

principal compta amb la seva millora de dany.

- **Apunts:** Els apunts són utilitzats únicament a mode de col·leccionisme. No tenen un ús concret més enllà de que són requerits per aconseguir el final definitiu del joc.

4.5.2. Objectes consumibles

La principal característica d'aquests objectes és que són d'un sol ús tot i que es pot disposar de més d'un objecte de cada tipus, pel que no són objectes que apareguin únicament una vegada durant el transcurs del joc. Aquest tipus d'objectes normalment s'adquireixen a la botiga, tot i que també poden ser trobats en els nivells. Els objectes disponibles són:

- **Pocions de vida:** És l'objecte consumible més comú, i en ser utilitzat recupera un 30% de la vida màxima del jugador.

- **Beguda energètica:** Quan es consumeix, proporciona un augment del 25% en el dany que provoca el protagonista durant 30 segons.

- **Bateria:** En ser consumit, regenera els escuts del vestit robòtic en la seva totalitat.

4.5.3. Objectes interactius

Els objectes interactius són aquells que permeten al jugador realitzar accions quan entra amb contacte amb ells. A continuació es mostra la llista dels objectes interactius disponibles en el joc:

- **Escales:** Escales de mà que permeten al personatge ascendir i descendir per elles.

- **Cartells:** Cartells amb informació per al jugador, com per exemple els controls del joc, pistes del camí a seguir, etc.

- **Portes:** Alguns edificis comptaran amb portes que habiliten la entrada a l'interior dels mateixos. Habitualment no requeriran cap condició especial per ser obertes, però en alguns casos és necessari l'ús de monedes.

4.6. Interfície gràfica d'usuari

La interfície gràfica d'usuari (GUI) és un element de gran importància a l'hora de transmetre informació a l'usuari. Aquesta interfície permet al jugador interactuar amb els esdeveniments que poden tenir lloc dins del joc i actua com a pont entre els dispositius físics com el teclat o un comandament de joc i el món del propi joc. Segons la capacitat de l'usuari per controlar el

personatge, es divideixen les finestres en dues categories: interfícies dins del joc i interfícies fora del joc.

La interfície fora del joc realitza la funció de proporcionar a l'usuari un entorn agradable i intuïtiu que el permeti navegar entre les diferents finestres del joc. És important que els menús siguin simples i el text que els acompanya descrigui de forma clara quina és la seva funció, d'aquesta manera l'usuari podrà transitar entre les escenes sense problemes.

Per altra banda, la interfície dins del joc té com a objectiu principal mostrar informació útil per al jugador de manera clara i directa, sense necessitat d'interrompre la partida. En aquest cas també és molt important que la informació estigui ben distribuïda per tal que l'usuari hagi de desviar la seva atenció sobre el joc el mínim possible.

Per començar a definir com serà la interfície gràfica d'usuari cal determinar quin serà el flux dels menús, obtenint així una representació clara de com el jugador es podrà moure a través de les diferents finestres del joc.

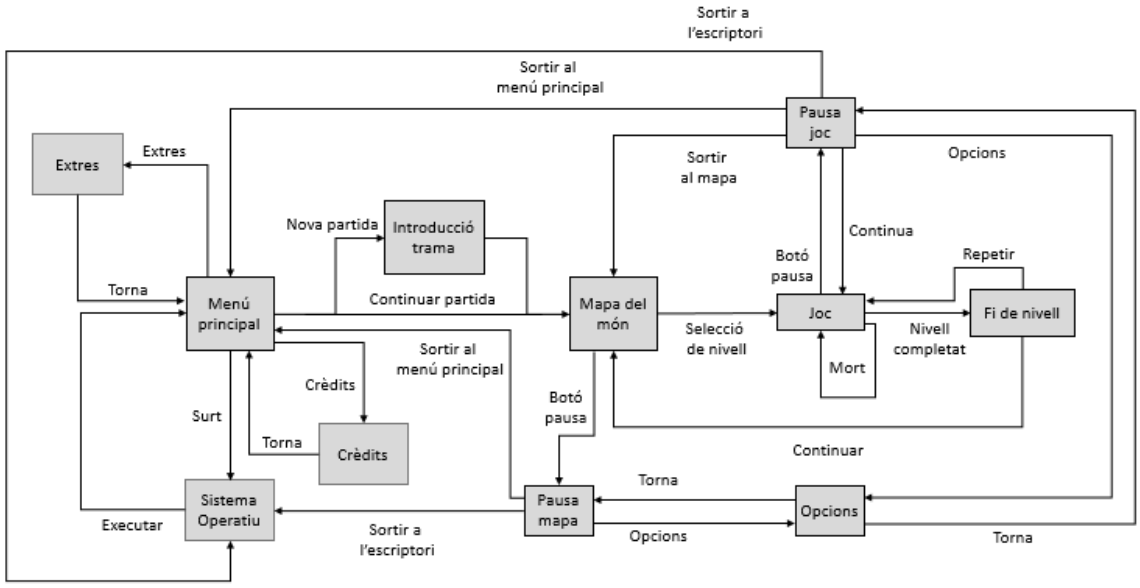

Figura 4.1. Diagrama de flux dels menús

El següent pas és definir quins elements estaran presents a cada una de les escenes, tal com es pot veure en els apartats posteriors.

4.6.1. Menú Principal

El menú principal és el primer contacte del jugador amb el joc un cop aquest sigui executat, i està format de botons de text que permeten escollir que fer a continuació.

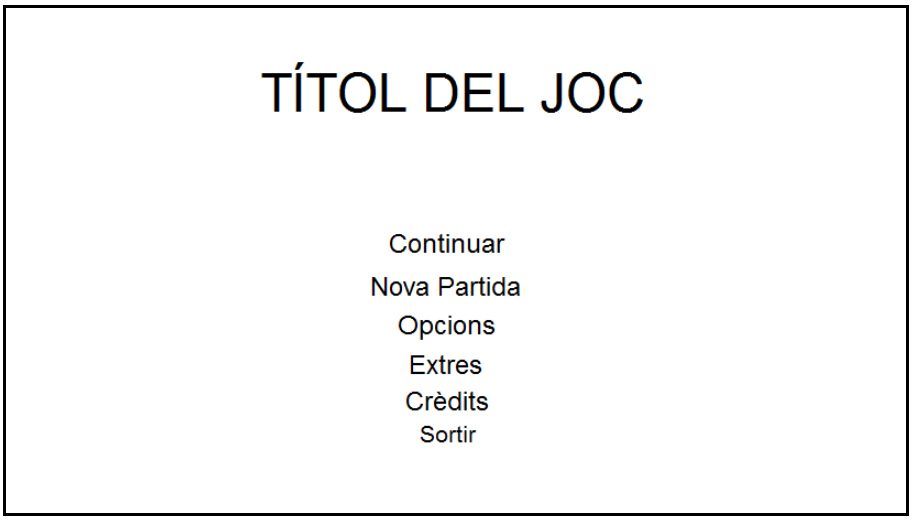

Figura 4.2. Menú principal

Com es pot comprovar a la ***Figura 4.2***, les funcions del menú són bastant clares:

- **Continuar:** El joc comença al punt on l'usuari va guardar partida en la seva darrera sessió de joc.

- **Nova partida:** S'esborra la partida actual i el joc comença des de l'inici. Per evitar que un possible error humà esborri la partida de forma inintencionada, cal afegir un menú de confirmació quan es selecciona aquesta opció que pregunti a l'usuari si realment vol esborrar la seva partida.

- **Opcions:** Carrega la finestra d'opcions, on el jugador pot modificar alguns paràmetres per adaptar el joc a les seves preferències.

- **Extres:** Aquest menú permet accedir a una secció amb afegits al mode principal del joc. Per exemple s'hi pot trobar una enciclopèdia d'enemics on es mostren totes les dades dels enemics vençuts o estadístiques de la partida actual.

- **Crèdits:** Mostra els crèdits i agraïments del joc.

- **Sortir:** Tanca el joc. Com en el cas del botó de *Nova partida,* es requerirà confirmació per sortir del joc per evitar possibles actes involuntaris.

4.6.2. Introducció a la trama

Quan el jugador inicia una nova partida, es mostra la finestra d'introducció a la trama. En aquesta finestra es posa a l'usuari en situació i s'explica quin és el context de la història que es desenvolupa un cop iniciada la partida.

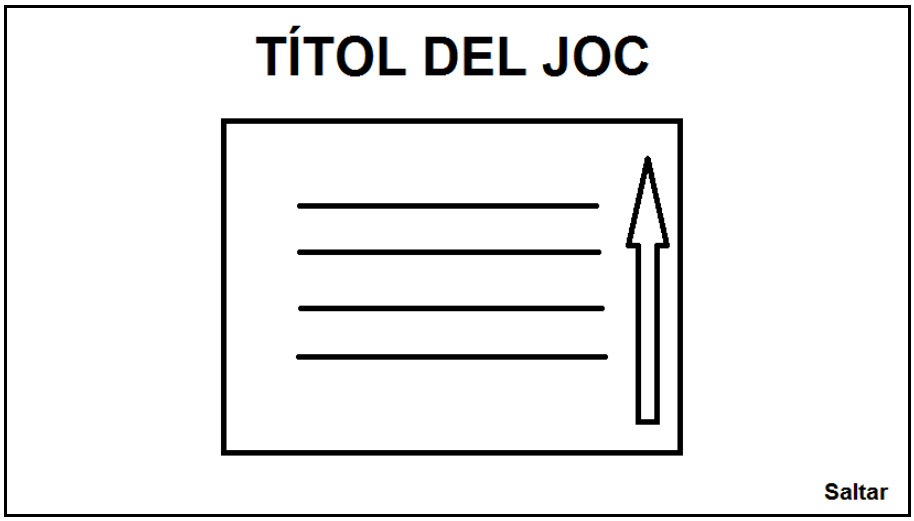

Figura 4.3. Introducció a la trama

Com s'aprecia a la ***Figura 4.3***, aquesta finestra mostra un text que apareix gradualment en pantalla a mesura que les línies es desplacen verticalment. També existeix la possibilitat d'ometre aquesta pantalla, per aquells usuaris que ja hagin jugat amb anterioritat i coneguin la trama.

4.6.3. Mapa del món

Una vegada començada l'aventura, a través del botons del menú principal *Nova partida* o *Continuar*, s'accedeix a un mapa del món que permet seleccionar entre els diferents nivells disponibles, així com la ciutat principal on es troben la botiga i la posada.

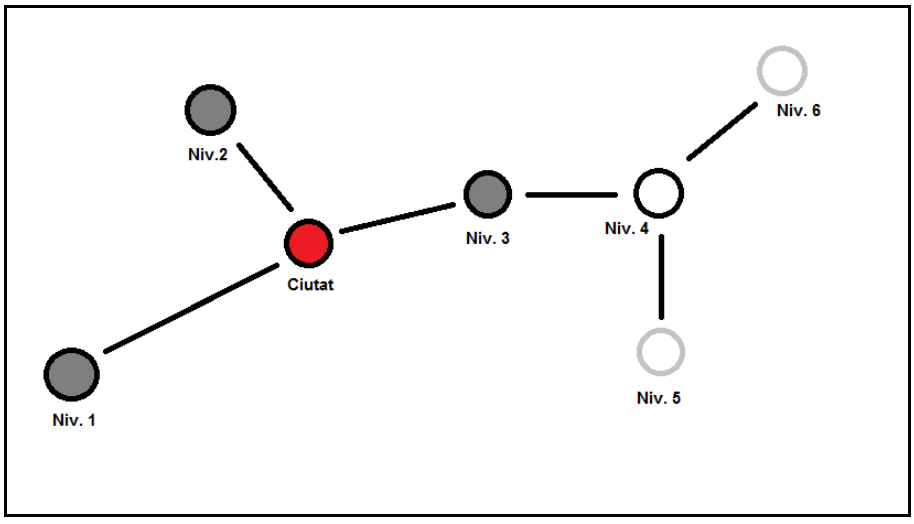

Figura 4.4. Mapa del món

Els nivells seran presentats de forma diferent en funció de l'estat de compleció o les

característiques que presentin. La classificació realitzada és la següent:

- **Nivells especials:** Són aquells que no contenen les mecàniques jugables característiques del joc i generalment no poden ser completats. L'únic nivell especial del joc és la ciutat, ja que s'utilitza per comprar objectes i recuperar salut.

- **Nivells completats:** Són els nivells que el jugador ha superat. Es poden tornar a jugar en qualsevol moment, però l'enemic final de cada nivell només apareix el primer cop que es juga. En el diagrama, l'usuari ha completat els nivells 1, 2 i 3.

- **Nivells desbloquejats:** Nivells als que l'usuari ha aconseguit accés però encara no ha superat. A mesura que s'avança en la trama es van desbloquejant els següents nivells. En el cas del diagrama mostrat, l'únic nivell desbloquejat és el nivell 4.

- **Nivells bloquejats:** El jugador no pot accedir a aquests nivells perquè encara no satisfà els requeriments per poder jugar-los. Habitualment l'únic requisit és haver superat el nivell immediatament anterior, tot i que poden haver casos especials.

El nombre de nivells mostrats a la ***Figura 4.4*** no es correspon amb el nombre de nivells del joc final, que es tracta a l'apartat de Nivells, disponible a la pàgina 27.

4.6.4. Interfície dins del joc

La distribució dels elements a la interfície dins del joc es mostren a la ***Figura 4.5***.

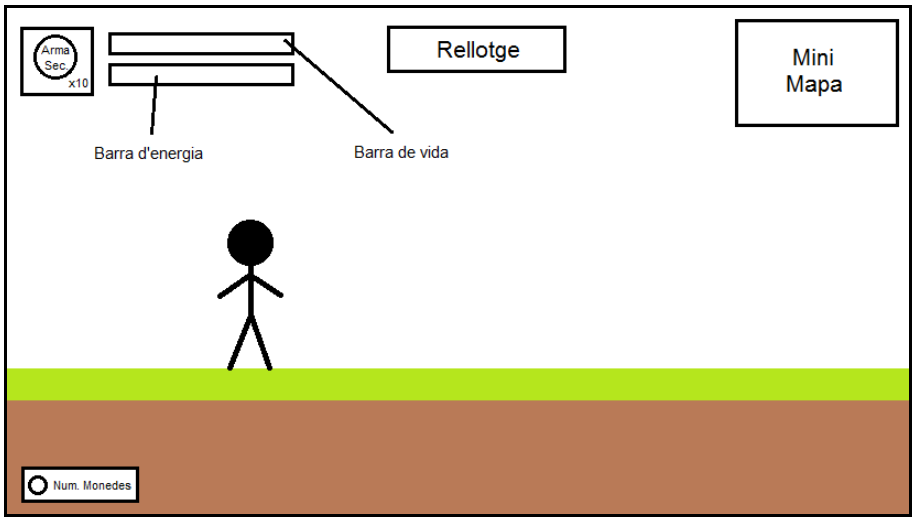

Figura 4.5. Interfície dins del joc

Els elements que composen aquesta interfície gràfica són:

- **Indicador d'arma secundària:** Situat a la part superior esquerra de la pantalla, indica

quina arma secundaria està activa en aquest moment i la munició disponible.

- **Barra de vida:** Mostra el nivell de salut del personatge.
- **Barra d'energia:** Mostra el nivell d'energia del personatge.
- **Rellotge:** Permet determinar el temps transcorregut des de l'inici del nivell.
- **Mini mapa:** Mostra quant s'ha avançat en el nivell i facilita la localització de sales que el jugador hagi pogut passar per alt.
- **Comptador de monedes:** Es troba a la part inferior esquerra i informa sobre la quantitat de monedes de que disposa el protagonista.

4.6.5. Pantalla fi de nivell

La pantalla de fi de nivell apareix cada vegada que el jugador completa un nivell, i mostra el temps emprat per superar el nivell i les monedes recollides. En cas que l'usuari superi el seu millor temps, s'indica amb un text que apareix al costat del temps.

Figura 4.6. Pantalla de fi de nivell

Addicionalment, a la part inferior dreta de la pantalla, es mostren els botons que permeten realitzar les accions de *Continuar,* que carrega el *mapa del món*, o *Repetir*, que permet a l'usuari tornar a jugar el nivell.

4.6.6. Menú de pausa

Un altre menú que suposa una part important de la interfície fora del joc és el menú de

pausa, que com el seu nom indica permet a l'usuari detenir l'acció de joc. Tot i que per accedir a ell cal estar dins del joc, mentre l'usuari estigui en aquest menú tot el món de joc es deté i no es pot jugar estant en ell, per aquest motiu no és considerat com a interfície dins del joc.

En realitat es tenen dos menús de pausa, un utilitzat durant les parts jugables i un altre per el mapa del món. Aquests dos menús de pausa són molt similars, essent la única diferència que el menú de pausa del mapa del món no té botó per retornar al mapa del món, ja que òbviament no és necessari.

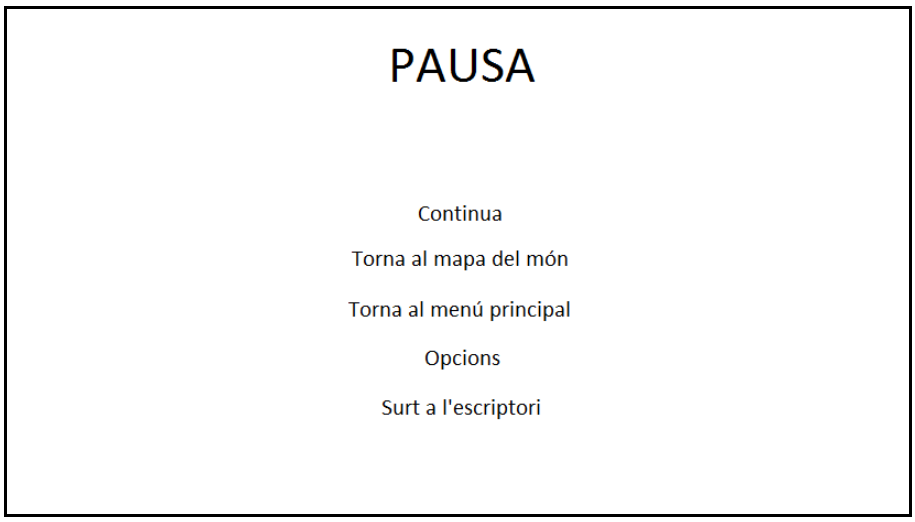

Figura 4.7. Menú de pausa

Tal com es mostra a la ***Figura 4.7***, el menú de pausa té les següents funcionalitats:

- **Continua:** Finalitza la pausa i permet continuar l'acció de joc.
- **Torna al mapa del món:** Abandona el nivell que s'està jugant i realitza la transició cap al mapa del món. El jugador perdrà tot el progrés no guardat.
- **Torna al menú principal:** De manera similar al botó *Torna al mapa del món,* carrega de nou el menú principal, perdent tot el progrés no guardat.
- **Opcions:** Mostra el menú d'opcions.
- **Surt a l'escriptori:** Tanca completament el joc. Com en casos anteriors, es perd tot el progrés no guardat

En el cas dels botons que fan perdre el progrés del jugador si no s'ha guardat prèviament, es mostrarà un altre menú de confirmació per evitar que el jugador perdi el seu avanç per causa d'un error.

4.6.7. Menú d'opcions

El menú d'opcions permet a l'usuari modificar una sèrie de paràmetres per tal de poder ajustar el joc a les seves preferències. Addicionalment, es mostra una descripció dels elements que composen la interfície dins del joc per si el jugador no comprèn el que veu en pantalla.

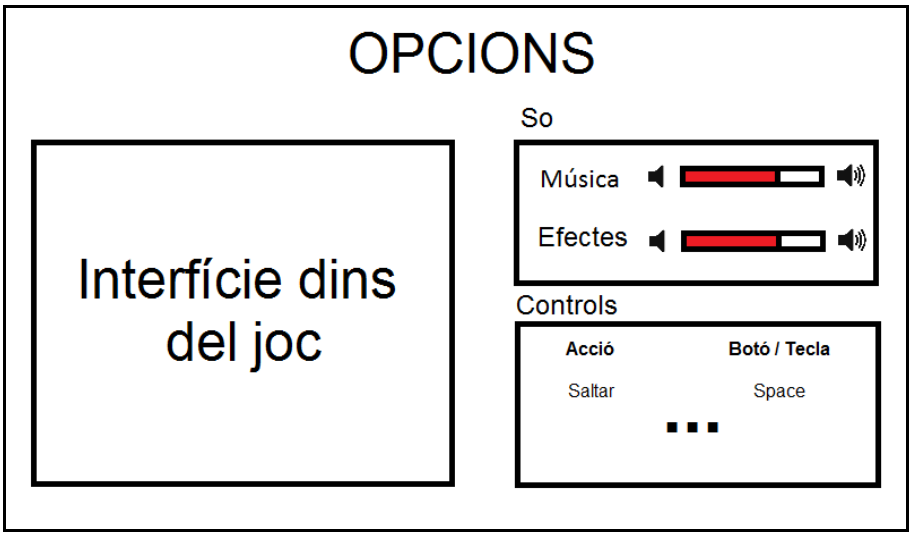

Figura 4.8. Menú d'opcions

Aquest menú disposa de tres funcions bàsiques, tal i com es veu a la ***Figura 4.8***:

- **Interfície dins del joc:** Es mostra una captura de pantalla del joc on s'explica de forma breu que indica cada element de la interfície dins del joc.

- **So:** Permet ajustar el volum dels sons, tant de la música de fons com dels efectes sonors.

- **Controls:** Permet personalitzar els controls del joc per adaptar-los a les preferències del jugador.

4.6.8. Crèdits

En aquesta escena es mostra qui ha realitzat cadascuna de les tasques en el desenvolupament del joc. En cas de que el nombre de participants sigui molt elevat, es pot implementar un text amb desplaçament vertical similar al de la *Introducció a la trama*.

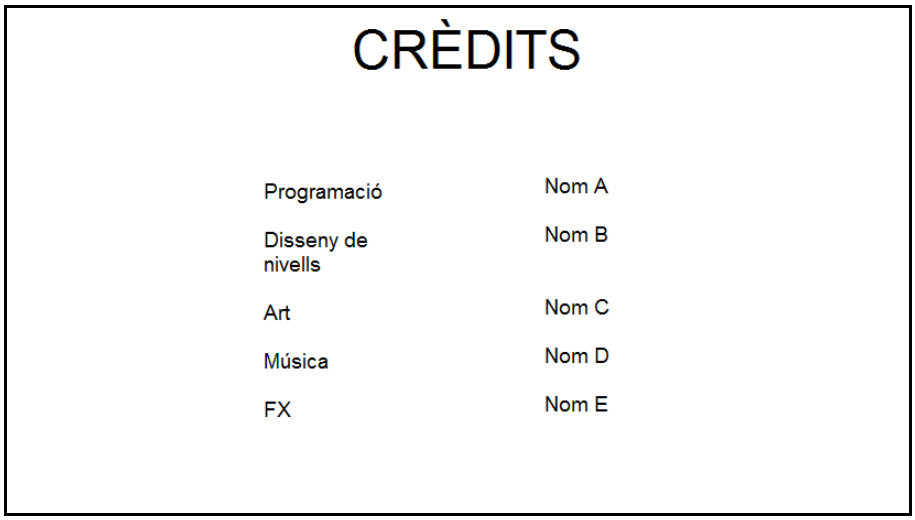

Figura 4.9. Crèdits

4.6.9. Extres

En aquesta finestra es troben els diferents continguts addicionals de que disposa el joc.

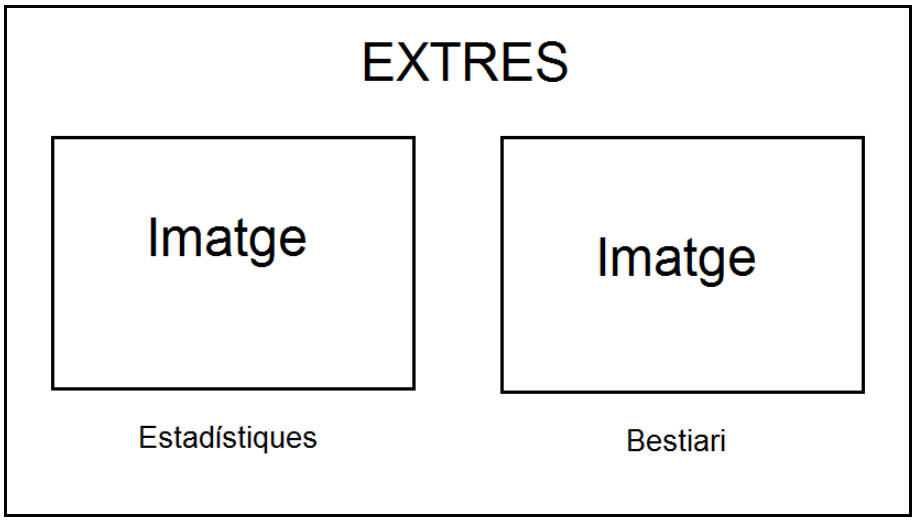

Figura 4.10. Extres

Com es veu a la ***Figura 4.10***, els extres disponibles són les estadístiques i el bestiari. Les estadístiques permeten a l'usuari revisar les dades de la seva partida com el temps de joc, les vegades de joc, etc. Pel que fa al bestiari, aporta tota la informació sobre els diferents enemics que el jugador ha derrotat durant el joc.

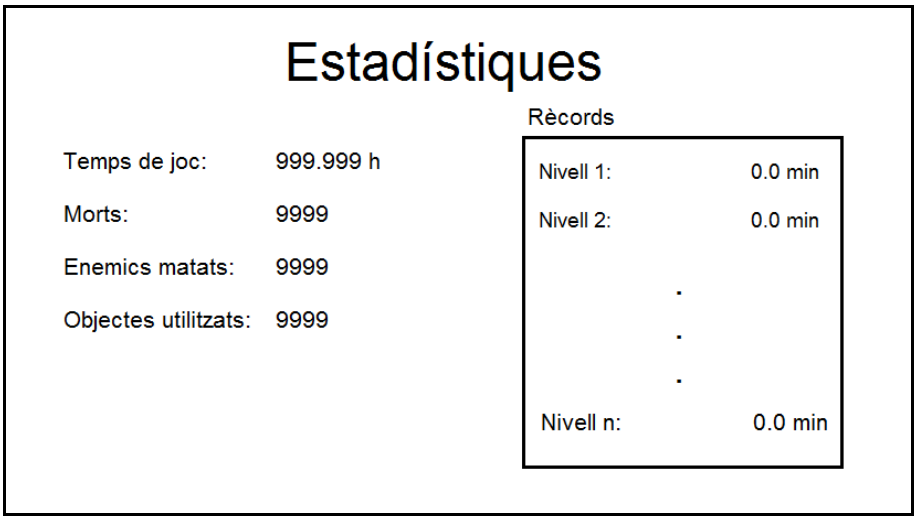

Figura 4.11. Estadístiques

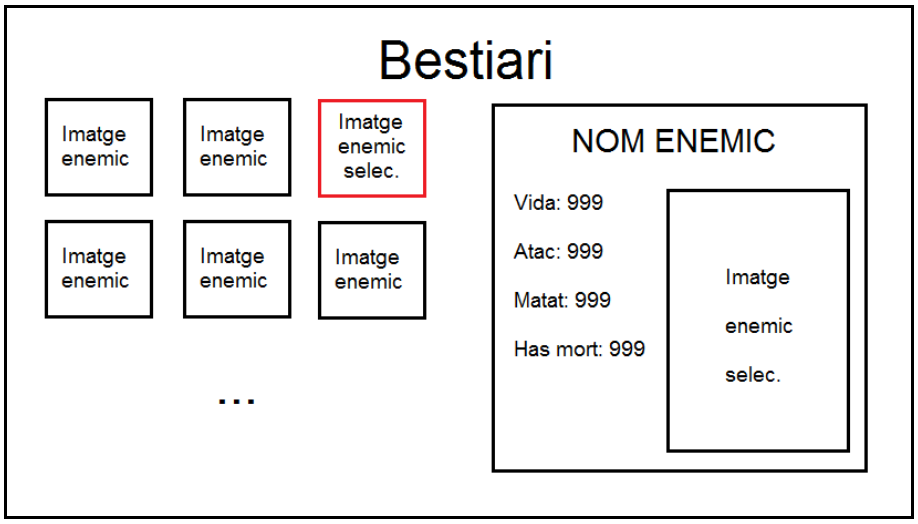

Figura 4.12. Bestiari

4.7. Nivells i trama

Com s'ha mencionat en apartats anteriors, el joc està dividit en nivells. L'objectiu en cada nivell és simplement arribar al seu final, però per aconseguir-ho caldrà superar els desafiaments plantejats a cada zona. Els únics enemics que han de ser derrotats obligatòriament són els enemics finals de cada nivell, tot i que vèncer als enemics comuns aporta un major nombre de monedes i permet una progressió del personatge més ràpida.

Els nivells serveixen com a connexió entre els diferents esdeveniments de la trama, i generalment després de derrotar a l'enemic final s'aconsegueix informació que permet al

protagonista descobrir més pistes sobre la localització de Grimgrorn.

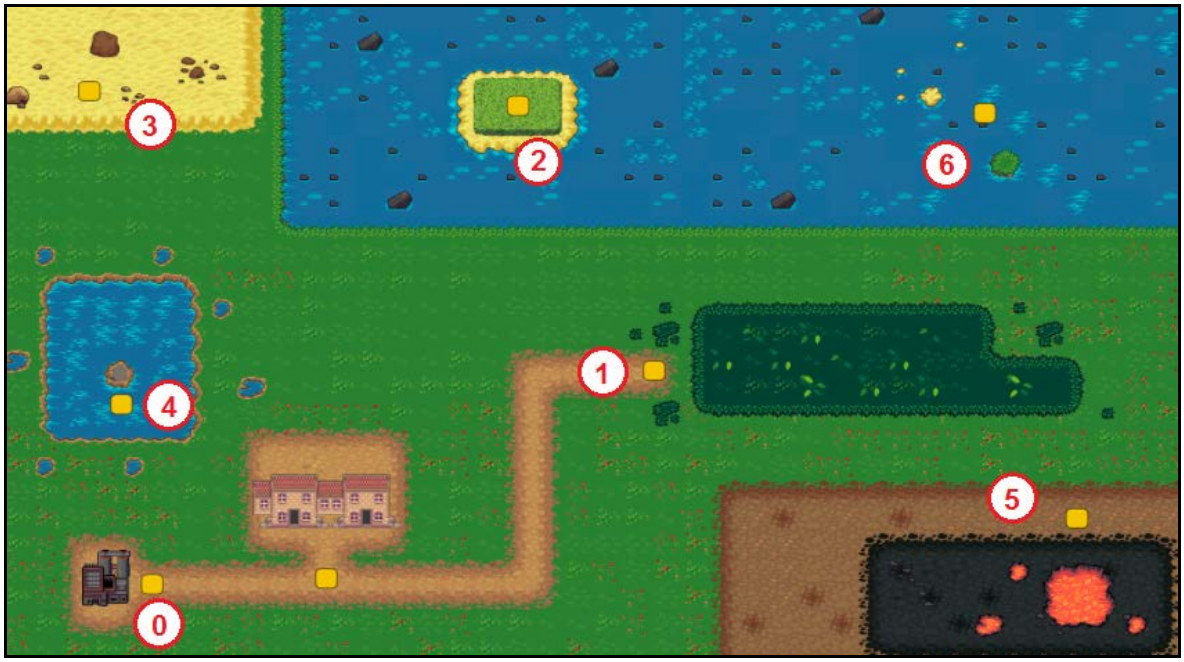

Figura 4.13. Mapa del món utilitzat al prototip amb la numeració dels nivells

En principi el joc disposa de 7 nivells, cadascun amb una ambientació diferent, tot i que en cas de realitzar un joc comercial es podria ampliar el nombre de nivells en funció del pressupost. En la **Figura 4.13** es mostra el mapa del món utilitzat al prototip realitzat durant el treball.

Els nivells inclosos al joc són:

- **Nivell 0:** Laboratori.
- **Nivell 1:** Bosc.
- **Nivell 2:** Illes.
- **Nivell 3:** Desert.
- **Nivell 4:** Llac.
- **Nivell 5:** Volcà.
- **Nivell 6:** Base secreta de Grimgrorn.
- **Poble:** Nivell no numerat, disponible en qualsevol moment després del nivell 0.

Els nivells no estan completament definits més enllà de la seva ambientació, excepte el laboratori (Nivell 0) i el Poble. En cas de realitzar el projecte complert caldria definir quins

esdeveniments ocorren en cada nivell en quant a la trama del joc.

El nivell del laboratori s'utilitza com a tutorial per introduir al jugador els moviments i les mecàniques del joc. Per aquest motiu, el disseny del nivell es basa en provocar situacions on el jugador faci ús de totes aquestes característiques. Pel que fa al poble, és un nivell al que sempre es pot accedir i conté 2 cases, la posada per recuperar salut i energia, i la botiga, per comprar munició, pocions, etc.

5. Implementació general del disseny

Una vegada es finalitza la fase de disseny del videojoc i estan definides totes les característiques que presentarà, cal implementar aquest disseny. En aquesta fase de desenvolupament cal decidir com es duran a terme les idees plantejades.

5.1. Motor de joc i llenguatge de programació

A l'hora d'implementar un videojoc, són necessàries múltiples funcionalitats com per exemple càlcul de físiques, representacions gràfiques interactives, reproducció de sons, etc. Es podria crear un motor de joc que implementés totes les característiques necessàries, però en l'actualitat existeixen una gran varietat de motors de joc que ofereixen moltes opcions i faciliten el desenvolupament de videojocs. Per les característiques del projecte, es va decidir utilitzar un motor amb llicència gratuïta per us personal o acadèmic, així com una bona documentació i funcionalitats 2D.

Les alternatives plantejades pel projecte són *Unity*, *Unreal Engine 4* i *GameMaker: Studio*, que s'adeqüen als requeriments desitjats en quant a plataformes. En tots tres motors es poden exportar els projectes a una gran multitud de plataformes com *Windows*, *Mac OS*, *Linux, Android*, *iOS* i jocs per navegador basats en *HTML5*.

Unity va començar com un motor de joc en 3D, però actualment s'han implementat una gran quantitat de funcionalitats per joc 2D. L'edició personal és gratuïta, i permet exportar els projectes en multitud de plataformes. Els llenguatges utilitzats per la programació de *scripts* són *C#* i *JavaScript*.

Unreal Engine 4 també és una molt bona opció ja que, de manera similar a *Unity*, es tracta d'un motor pensat per 3D que poc a poc està incorporant un gran nombre d'eines per desenvolupar jocs en 2D. *Unreal Engine 4* és un motor molt potent i actualment és totalment gratuït si no es generen ingressos amb el joc, en cas contrari cal pagar un 5% dels beneficis obtinguts. El llenguatge a utilitzar en aquest motor és *C++*.

Pel que fa a *GameMaker: Studio*, és un motor per a jocs 2D força senzill i intuïtiu, ideal per endinsar-se en el món del desenvolupament de videojocs per primera vegada. El principal problema és que per exportar els jocs a plataformes diferents a *Windows*, cal comprar mòduls que es venen per separat, per tant es descarta. Utilitza un llenguatge de programació propi anomenat *Game Maker Language (GML)*.

Finalment es va prendre la decisió d'utilitzar *Unity*, principalment per la gran documentació i comunitat de que disposa, així com també la possibilitat d'utilitzar el llenguatge *C#*, bastant més senzill que *C++* en base als coneixements adquirits durant el grau, que han estat amb llenguatge *Python*.

5.2. Treballant amb Unity

Unity disposa de nombroses característiques que faciliten el desenvolupament de videojocs. En aquest apartat s'inclou una breu descripció dels recursos més utilitzats en el desenvolupament del prototip, per tal d'introduir tots els conceptes necessaris per entendre la implementació de cada un dels elements del joc.

5.2.1. Escena (Scene)

L'escena conté els objectes del joc i pot ser utilitzada per a qualsevol cosa, des de per mostrar el menú principal fins a un nivell del joc. Es pot definir com el món espacial en el que es desenvolupa el joc.

Per exemple, es pot construir cada nivell com una escena per separat, ja que és senzill canviar entre una escena i una altra mitjançant codi. En cada escena es situen tots els objectes necessaris per aquell nivell com el personatge, els enemics, els objectes, les trampes, etc.

5.2.2. GameObjects

Els *GameObjects* són el tipus d'objecte més important en *Unity*. Tots els elements del joc són *GameObjects* (els personatges, els objectes, els escenaris, la interfície d'usuari, etc.), per aquest motiu cal comprendre correctament quina és la seva funció.

Tot i que és veritat que tots els elements del joc són d'aquest tipus, un *GameObject* per si sol no realitza cap acció, si no que actua com a un contenidor de components, que són els que realitzaran les accions requerides. Per aquest motiu cada *GameObject* pot ser totalment diferent dels altres i poden realitzar una gran varietat d'accions dins del joc, ja que *Unity* incorpora un gran nombre de components preparats per a ser utilitzats sense dificultats. L'únic component comú en tots els objectes de tipus *GameObjects* és el component *Transform*, que determina la posició, la rotació i l'escala de l'objecte en els eixos X, Y i Z.

Una altra característica dels *GameObjects* és que es poden definir relacions de pare / fill entre objectes, on els valors del component *Transform* del fill són relatius al pare.

A mode d'exemple, a continuació es mostra la diferència entre dos elements ben diferenciats entre sí, tot i ser ambdós *GameObjects*.

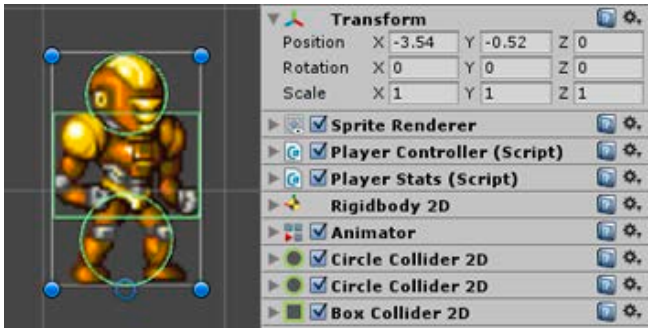

Figura 5.1. GameObject del protagonista

En el cas de la ***Figura 5.1***, que mostra el *GameObject* que conforma el personatge principal, es poden observar els components que té adjunts:

- ***Transform*:** Determina la posició, la rotació i l'escala de l'objecte en els eixos X, Y i Z.
- ***Sprite Renderer*:** Mostra per pantalla la imatge associada al personatge en cada un dels fotogrames.
- ***Player Controller*:** *Script* que controla els moviments i accions del personatge (caminar, saltar, disparar, etc.)
- ***Player Stats*:** *Script* que emmagatzema les dades del personatge com la vida, les monedes, etc.
- ***Rigidbody 2D*:** Permet que el *GameObject* sigui controlat segons el motor de físiques incorporat a *Unity*. Com el seu nom indica és un cas particular per 2D, reduint el moviment als plans X i Y i la rotació al pla Z.
- ***Animator*:** Controla les animacions del personatge, que han de ser definides prèviament, determinant quin *sprite* s'ha de mostrar per pantalla a cada fotograma.
- ***Circle / Box Collider 2D*:** Permet definir col·lisionadors que es poden referenciar posteriorment en *scripts*, facilitant la realització d'accions en entrar en contacte amb altres col·lisionadors.

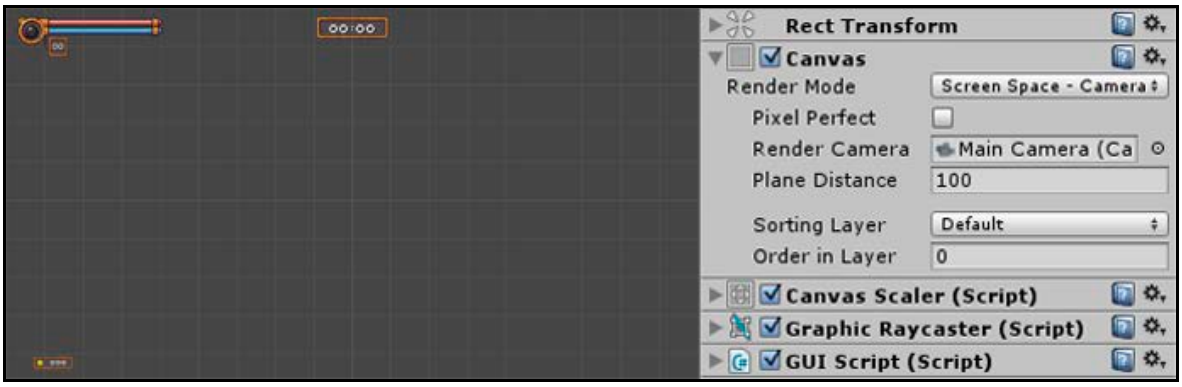

Figura 5.2. GameObject del canvas

En aquest segon exemple es parla del *canvas*, que gestiona tot allò que es mostra per sobre de la pantalla de joc, en aquest cas la interfície d'usuari. Com es mostra a la ***Figura 5.2***, aquest *GameObject* té adjunts els següents components:

- ***Rect Tansform*:** És un component similar a *Transform*, però en lloc de situar un punt en l'espai, determina el rectangle on es trobaran els elements de la interfície.

- ***Canvas*:** Determina l'espai on es representen i renderitzen tots els elements de la interfície gràfica. Tots els *GameObjects* relacionats amb la interfície han de ser fills del *GameObject* que contingui aquest component.

- ***Canvas Scaler*:** Com el seu nom indica, controla l'escala de tots els elements de la interfície.

- ***Graphic Raycaster*:** s'utilitza per emetre raigs contra el *canvas*, per tal de detectar si algun element de la interfície ha col·lidit amb un dels *GameObjects* de la escena. S'utilitza per configurar el comportament de la interfície en cas que aquestes col·lisions es produeixin, per defecte els objectes del *canvas* es renderitzen a la capa més externa.

- ***GUI Script*:** *script* que controla els elements variables en la interfície, com el rellotge, el comptador de monedes, etc.

Com es pot comprovar, els *GameObjects* són la base del desenvolupament de videojocs amb *Unity*.

5.2.3. Components més comuns

Existeixen molts components incorporats a Unity, però alguns d'ells han estat utilitzats al projecte amb molta més assiduïtat que d'altres. L'objectiu d'aquest apartat es descriure els

components més habituals ja que la majoria de *GameObjects* faran ús d'ells, d'aquesta manera serà més senzill analitzar les particularitats de cada cas. Per aquest motiu, els components amb funcionalitats molt específiques es descriuran a l'apartat que expliqui la implementació de cada *GameObject* en concret.

5.2.3.1. Transform

Com s'ha mencionat amb anterioritat, aquest component és el més bàsic, doncs tots els *GameObjects* l'incorporen per defecte. Els paràmetres del component es mostren a la ***Figura 5.3***.

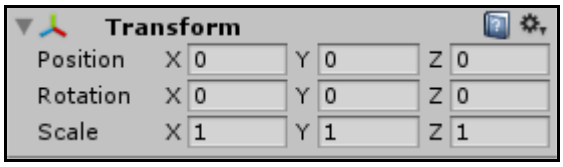

Figura 5.3. Paràmetres del component Transform

5.2.3.2. Sprite Renderer

El *Sprite Renderer* s'encarrega de mostrar un *sprite* a l'escena actual, per tant tots els *GameObjects* que hagin d'aparèixer en pantalla i que parteixin d'una imatge inclouran aquest component.

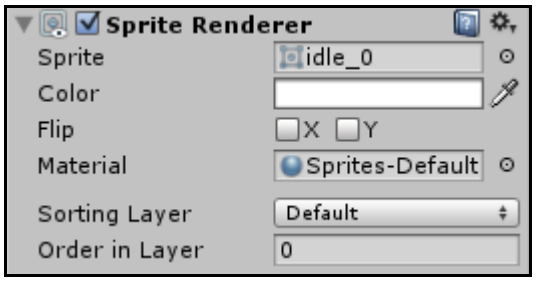

Figura 5.4. Paràmetres del component Sprite Renderer

Els paràmetres que es poden modificar en aquest component són:

Paràmetre	Funció
Sprite	Imatge amb el *sprite* a mostrar per pantalla
Color	Color aplicat al *sprite*. Aquest color s'aplica multiplicat al color de base, per tant el color blanc no canvia el color original.
Flip	Voltejar el *sprite* original, tant en l'eix X com Y.

Material	Material utilitzat per renderitzar el *sprite*.
Sorting Layer	Capa en que es troba aquest *sprite*. D'aquesta manera és possible assignar prioritats en l'ordre de renderitzat.
Order in layer	Ordre de renderitzat del *sprite* dins la seva capa.

5.2.3.3. Rigidbody 2D

Aquest component fa que el *GameObject* al que està adjunt es comporti segons el motor de físiques de *Unity*. És un cas particular del component *Rigidbody* on el moviment està restringit al pla XY i la rotació al voltant de l'eix Z. Si s'aplica un *Collider 2D* al mateix *GameObject*, es possibilita que les col·lisions entre *GameObjects* es comportin segons el motor de físiques, aconseguint un resultat més realista.

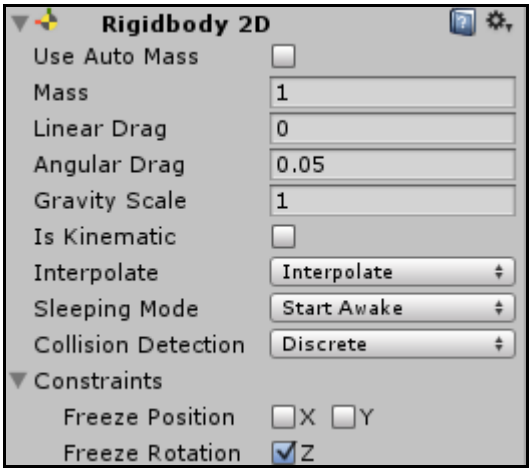

Figura 5.5. Paràmetres del component Rigidbody2D

Els paràmetres que conté aquest component són:

Paràmetre	**Funció**
Use Auto Mass	Si està activat, calcula la massa en funció dels *Colliders* adjunts
Mass	Massa del sòlid rígid
Linear Drag	Coeficient d'arrossegament que afecta al moviment
Angular Drag	Coeficient d'arrossegament que afecta a la rotació
Gravity Scale	El grau d'afectació de la gravetat sobre el sòlid rígid

Is Kinematic	Si està activat el sòlid rígid pot ser desplaçat per forces i col·lisions
Interpolate	De quina manera s'interpola entre les actualitzacions de la física
Sleeping Mode	Determina com l'objecte "dorm" per estalviar processos quan està inactiu
Collision detection	Mètode de detecció de col·lisions. En discret només es detecten si col·lideix quan s'actualitzen les físiques i en continu es detecten també entre actualitzacions
Freeze Position	Restringeix el moviment en X, en Y o en totes dues direccions
Freeze Rotation	Restringeix la rotació al voltant de Z

5.2.3.4. Collider 2D

El component *Collider 2D* permet enregistrar les col·lisions entre objectes que tinguin adjunts aquest tipus de component. Existeixen quatre tipus de *Collider 2D, Circle, Box, Polygon* i *Edge*. Els més utilitzats són els de tipus *Circle* i *Box*, ja que en poder utilitzar-se més d'un col·lisionador a l'hora en el mateix *GameObject* és senzill cobrir la totalitat del sprite consumint menys recursos que amb un *Polygon*. A continuació es mostra una comparativa entre els diferents tipus de *Collider 2D*.

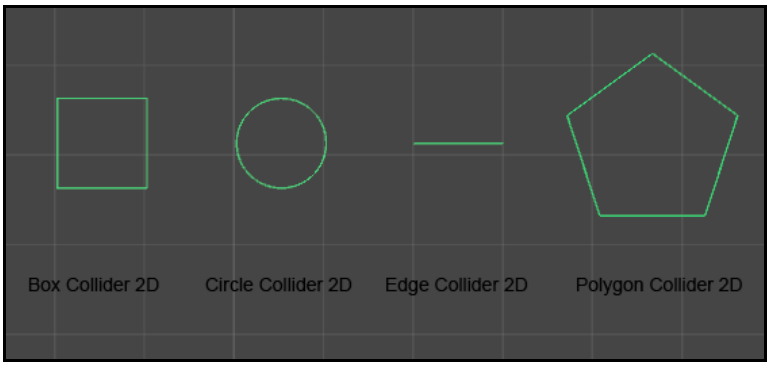

Figura 5.6. Tipus de Collider 2D

Els paràmetres de cada tipus són diferents per a poder definir la geometria desitjada, però existeixen tres paràmetres comuns a tots els tipus de *Collider 2D*.

Paràmetre	**Funció**
Material	Permet utilitzar un *Physics Material* al col·lisionador que controla propietats com la fricció i el rebot

Is Trigger	Si està activat, el col·lisionador actua com a desencadenant d'alguna acció
Used by Effector	Determina si el col·lisionador és utilitzat pel component *Effector* adjunt al mateix *GameObject*

5.2.3.5. Animator

El component *Animator* s'utilitza per assignar animacions a un *GameObject*. Aquest component requereix tenir associat un controlador d'animacions, que defineix les animacions a utilitzar i en quines circumstàncies s'han de realitzar les transicions entre elles. En apartats posteriors es detalla el funcionament dels controladors d'animacions.

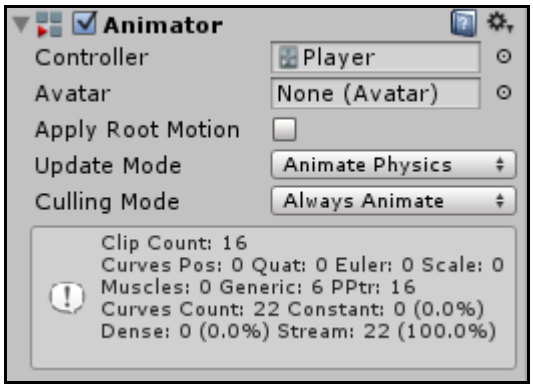

Figura 5.7. Paràmetres del component Animator

Com es pot veure a la ***Figura 5.7***, a la part inferior es mostra la informació sobre el controlador d'animacions associat al component. Els paràmetres del component Animator són els següents:

Paràmetre	**Funció**
Controller	El controlador d'animacions associat al *GameObject*
Avatar	L'Avatar del *GameObject* (només per humanoides 3D)
Apply Root Motion	Quan està activat, es controla el moviment del personatge / objecte a través de les animacions en comptes de controlar-ho mitjançant un script
Update Mode	Permet seleccionar de quina manera s'actualitza el controlador d'animacions
Culling Mode	Permet seleccionar de quina manera es tallen les animacions

5.2.3.6. Script

El component *Script* és un dels més importants, doncs permet adjuntar codi a un *GameObject*, que pot ser en llenguatge *C#* o *JavaScript*. Com s'ha mencionat amb anterioritat, al projecte s'ha utilitzat *C#*.

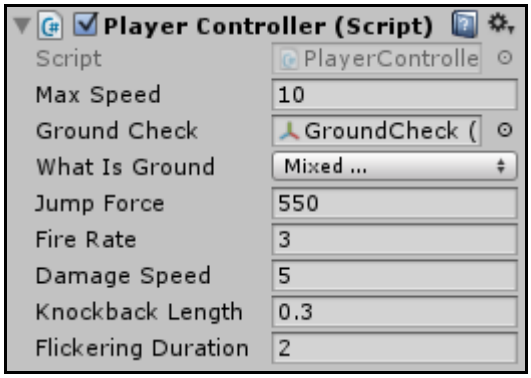

Figura 5.8. Exemple de component Script

Aquest component té únicament un paràmetre per defecte, que és el *script* a utilitzar. Els altres paràmetres són definits per l'usuari, doncs només cal declarar com a públiques les variables desitjades per a que apareguin com a paràmetre i es pugui modificar el seu valor des de la interfície de *Unity*. En l'exemple presentat a la ***Figura 5.8*** es poden veure els diversos paràmetres de que es disposen en aquest cas.

5.2.4. Classe MonoBehaviour i funcions predefinides

Quan es programa un *script* creat des de *Unity*, es disposa per defecte d'una classe anomenada *MonoBehaviour*, que conté una sèrie de funcions que resulten de gran utilitat per implementar qualsevol comportament desitjat en el joc. Sempre que es crea un script nou, *Unity* defineix una classe nova específica per el script, que hereta totes les característiques de la classe *MonoBehaviour*.

Existeixen una gran varietat de funcions i mètodes inclosos dins *MonoBehaviour* però en aquest apartat s'explicaran les més utilitzades al projecte. Les funcions més específiques seran detallades quan s'expliqui alguna característica que en faci ús.

- ***Start:*** La funció *Start* s'executa en el primer fotograma de la escena, abans que qualsevol de les altres funcions. Habitualment s'utilitza per inicialitzar variables.

- ***Update:*** *Update* s'executa cada fotograma, i s'utilitza per a implementar qualsevol comportament. El temps d'execució entre un Update i el següent pot variar ja que

també ho fan els recursos utilitzats en cada fotograma, per aquest motiu no es sol utilitzar per fer càlculs relacionats amb el motor de físiques.

- ***FixedUpdate:*** *FixedUpdate* és una funció similar a Update però a diferència d'aquesta, que s'executa a cada fotograma, *FixedUpdate* s'executa just abans que el motor de físiques actualitzi l'estat de tots els *GameObjects* que incorporin físiques. D'aquesta manera, l'interval entre un *FixedUpdate* i el següent és molt més regular, el que el fa convenient per programar accions relacionades amb les físiques.

- ***OnCollisionEnter2D:*** Aquesta funció s'executa quan el col·lisionador d'un altre *GameObject* entra en contacte amb el col·lisionador del *GameObject* que conté el *script*. També existeixen les funcions *OnCollisionStay2D* i *OnCollisionExit2D* que s'executen mentre la col·lisió continua i al sortir d'aquesta, respectivament. Aquestes funcions són especialment útils per definir la interacció entre objectes i, com es pot observar pel seu nom, són un cas particular per col·lisionadors 2D.

5.2.5. Input

En el cas de jocs per computador, *Unity* suporta com a dispositius d'entrada el teclat, el ratolí i joysticks / comandaments. A la interfície gràfica de *Unity*, mitjançant el *InputManager*, es poden configurar els diferents botons i eixos que s'utilitzaran en el joc. Tot botó pot ser tractat com un eix, però generalment si es vol el comportament d'un botó es defineix únicament la part positiva.

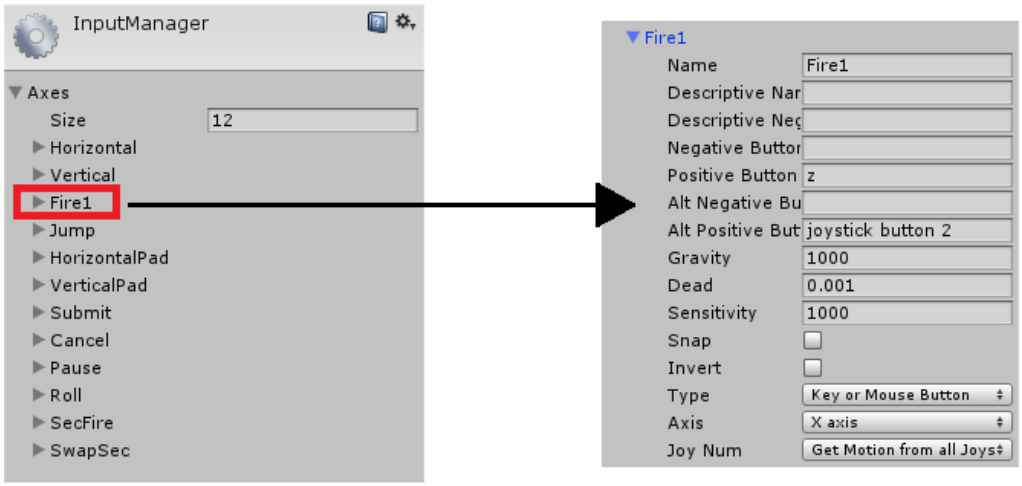

Figura 5.9. InputManager de Unity

La característica més important del *InputManager* és que permet configurar els controls per a qualsevol dispositiu d'entrada suportat i permet accedir-hi de manera senzilla quan s'està escrivint un script mitjançant la classe *Input*, que permet determinar quan un botó específic

es pressiona o deixa de ser pressionat.

Figura 5.10. Exemple del valor de l'eix horitzontal, en aquest cas polsat amb una intensitat del 80%

En el cas dels eixos, com l'horitzontal o el vertical, la classe *Input* conté una funció que permet determinar amb quina intensitat s'està pressionant l'eix, en un rang de -1 a 1. Aquesta informació resulta útil a l'hora de calcular la velocitat de moviment de personatges jugables.

Quan s'està escrivint un *script*, s'accedeix als botons i eixos pel seu nom independentment de quina sigui la configuració de botons del dispositiu físic, el que permet que el codi sigui robust en front de canvis en la distribució botons.

5.2.6. Prefabs

En el desenvolupament d'un videojoc, moltes vegades cal reutilitzar objectes o enemics durant el transcurs d'un nivell. Una solució seria realitzar un duplicat *GameObject* desitjat, però es crea el problema de que l'original i la còpia són editats de forma separada, cosa que generalment no resulta convenient.

Per aquest motiu, *Unity* permet treballar amb el que s'anomenen *prefabs*, que consisteix en crear un *GameObject* que fa la funció de plantilla de totes les seves instàncies. D'aquesta manera, s'aconsegueix que totes les instàncies canviïn les seves propietats modificant únicament el *GameObject* que actua com a *prefab*.

Addicionalment, es poden modificar individualment les instàncies de forma que algun paràmetre o component sigui diferent als del *prefab*, i també es permet aplicar els canvis realitzats a una instància sobre el *prefab* modificant totes les altres instàncies de manera automàtica. Per tant, la utilització de *prefabs* aporta una gran flexibilitat a l'hora de reutilitzar els recursos.

5.2.7. Etiquetes

Unity incorpora un sistema d'etiquetes que permet classificar els *GameObjects* segons els criteris que el creador consideri. En el cas del projecte, les etiquetes han resultat especialment útils a l'hora d'implementar la interacció entre *GameObjects*, definint com es

produeixen en funció dels tipus d'objectes.

Per classificar un *GameObject* sota una etiqueta en concret, només cal editar l'objecte a Unity i seleccionar l'etiqueta que es vol aplicar sobre ell. Un *GameObject* només pot tenir una etiqueta.

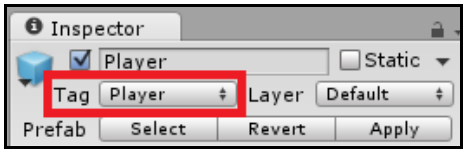

Figura 5.11. Exemple d'etiqueta en Unity

Aquesta etiqueta és fàcilment accessible quan s'escriu el codi, i es pot utilitzar, per exemple, per detectar si l'objecte amb el que s'ha col·lidit és un enemic o simplement per fer referència a objectes concrets dins del codi.

5.3. Recursos artístics

En el món dels videojocs existeixen unes possibilitats molt amplies a l'hora de mostrar el producte per pantalla, fet que provoca que la metodologia de disseny dels recursos artístics utilitzats variï segons el projecte. En general, aquests recursos artístics es poden classificar en dues grans categories, recursos 3D i 2D.

Els recursos 3D són creats normalment amb software de modelat com per exemple Maya, 3ds Max o Blender. Els principals tipus de recursos 3D són:

- **Models:** Els personatges i objectes utilitzats en el joc són modelats amb software com els citats anteriorment.

- **Animacions:** Les animacions del personatges habitualment es realitzen amb el mateix software amb que es modelen. L'animació de personatges és una part molt important en l'aspecte d'un videojoc, especialment en personatges humanoides, ja que els usuaris tenen experiència en el mon real i és fàcil detectar una mala animació, fet que pot empitjorar l'experiència de joc.

- **Entorn 3D:** Un altre element de gran importància en els jocs 3D. Els entorns 3D en un joc han de representar mons creïbles per tal d'aconseguir una major immersió del jugador.

- **Il·luminació:** La utilització d'una il·luminació i colors adequats pot millorar molt

l'aspecte gràfic en un videojoc, aconseguint així un producte més atractiu per l'usuari.

Addicionalment és necessari que els recursos siguin lleugers, doncs a mesura que augmenta la complexitat i similitud amb la realitat d'aquests recursos augmenta també la potència requerida per córrer el joc de forma fluida. El principal punt a favor de realitzar un joc en 3D és la possibilitat de recrear entorns i personatges molt més fidels a la realitat que si s'utilitzen recursos 2D.

Pel que fa als recursos 2D, són utilitzats tant en jocs 2D com 3D i existeix una gran varietat de tipus de recursos:

- **Art conceptual:** Es composa de diferents esbossos, dibuixos a mà alçada i guions il·lustrats que mostren els aspectes clau del videojoc, com els seus personatges o les principals mecàniques jugables. L'art conceptual és utilitzat per transmetre els conceptes principals de que disposarà el joc un cop finalitzat.

- ***Sprites*:** Són la representació 2D de personatges o objectes. En el cas de necessitar animacions, cal realitzar un *sprite* per cada fotograma clau de l'animació.

- **Fons dels nivells, terrenys i *tilesets*:** Són els blocs de construcció per qualsevol joc bidimensional. En particular, els *tilesets* resulten de gran importància ja que permeten estalviar recursos en la generació de nivells.

- **Interfície gràfica d'usuari:** En qualsevol videojoc, ja sigui 2D o 3D, cal mostrar al jugador una sèrie d'informació en pantalla com per exemple la puntuació o vides, així com menús i diferents pantalles de presentació. Per aconseguir això, cal dissenyar tots els elements que formaran part de la interfície d'usuari.

- **Textures i materials:** Les textures són utilitzades en personatges o objectes 3D, ja que permeten aconseguir un major detall als models i que la superfície d'aquests pugui interactuar amb la il·luminació.

En el cas del projecte, com s'ha mencionat amb anterioritat, tots els recursos artístics utilitzats són en 2D, treballant amb *sprites* per als personatges i objectes. Per els nivells s'han utilitzat *tiles*.

Els recursos artístics requereixen de personal especialitzat, ja que l'aspecte visual del videojoc és de vital importància a l'hora de causar interès en els potencials clients. Per les característiques d'aquest projecte, emmarcat dins del *Treball de Final de Grau* d'una titulació d'Enginyeria, i tenint en compte que el resultat obtingut serà un prototip, s'ha decidit que els recursos artístics utilitzats no siguin originals pel joc si no d'us lliure, creats per la comunitat.

El fet d'utilitzar recursos amb llicència d'us oberta estalvia temps i recursos a l'hora de realitzar un prototipatge, el que permet fer més èmfasi en la programació de les mecàniques de joc. En els següents apartats es mostra com s'ha dut a terme la implementació dels diferents tipus de recursos necessaris a *Unity*.

5.3.1. Personatges i objectes

En aquest apartat es mostra la metodologia utilitzada a l'hora de tractar els personatges i objectes per tal d'aconseguir una correcta implementació a *Unity*. Els personatges i objectes són recursos 2D del tipus *sprite*.

En primer lloc es crea el *sprite* amb un software de tractament d'imatge, realitzant tots i cada un dels fotogrames clau de l'animació en cas de requerir moviment. Tots els *sprites* es guarden en una sola imatge anomenada *sprite sheet*, que conté tots els *sprites* que conformen les animacions del personatge o objecte. En el cas d'estudi, els *sprites* utilitzats són d'us lliure, pel que s'ha treballat directament amb el *sprite sheet*.

Figura 5.12. Sprite sheet del personatge principal

Per comoditat, es crea un nou *sprite sheet* per a cada animació, que posteriorment és importat a *Unity*. Mitjançant l'editor de *sprites* incorporat al propi software, es defineixen les coordenades de la caixa que conté cada un dels *sprites* inclosos a la fulla, així com el punt de pivot que *Unity* utilitza com a origen de coordenades quan es reprodueix l'animació.

Figura 5.13. Editor de sprites de Unity

A la ***Figura 5.13*** es mostra el tractament de l'animació del personatge caminant. Com es pot observar, cada sprite té definida la seva pròpia caixa i, en el cas del *sprite* seleccionat, també es pot veure el seu punt de pivot que es troba a la part inferior de la capsa alineat al centre. Per visualitzar o modificar el punt de pivot de cada sprite, cal seleccionar el *sprite* desitjat.

Un cop es tenen els diferents *sprites* definits, cal crear clips animats indicant el temps durant el que es reproduirà cada sprite. Per realitzar aquesta tasca s'utilitza l'editor d'animacions de *Unity*.

Figura 5.14. Editor d'animacions de Unity

Quan s'han definit totes les animacions necessàries, es controlen mitjançant el controlador d'animacions de *Unity* anomenat *Mechanim*, que permet definir la manera en que es produeixen les transicions entre animacions mitjançant una màquina d'estats.

Per definir de quina forma es realitzen les transicions, en primer lloc cal definir les variables que tindrà el controlador, que són fàcilment modificables des dels *scripts*. *Unity* permet definir al controlador variables de tipus enter, real, booleà o *Trigger,* un booleà especial que es reinicia des del controlador quan es produeix una transició.

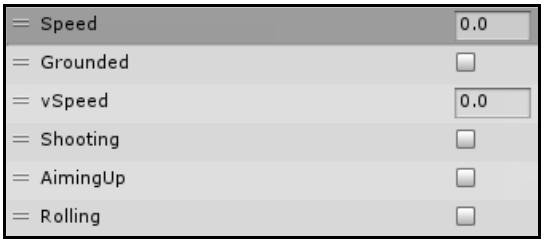

Figura 5.15. Exemple de variables a un controlador d'animacions

Un cop es tenen les variables necessàries, es configura des de la interfície de *Unity* quines són les condicions que s'han de donar per a que es produeixi la transició entre un estat i un altre. En l'exemple que es mostra a la ***Figura 5.16*** es produeix la transició quan la variable del controlador anomenada *Rolling* pren com a valor *true.*

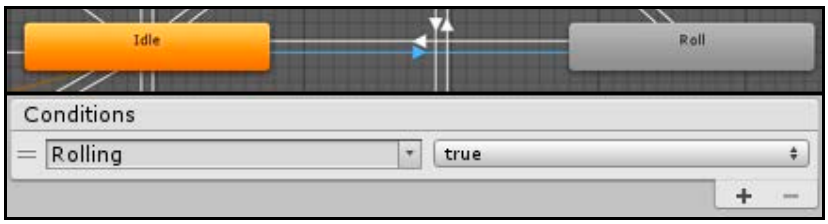

Figura 5.16. Exemple de transició entre estats

El controlador també permet configurar quin estat es reproduirà per defecte quan l'objecte entri en escena, que es mostra en color taronja. Addicionalment, es disposa de 3 estats que són inclosos per defecte, que són *Any State, Entry*, i *Exit*.

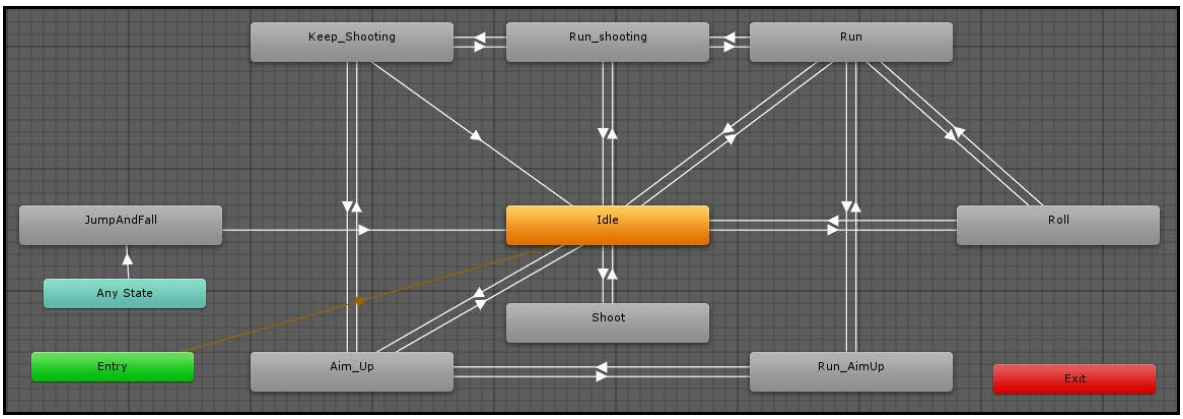

Figura 5.17. Exemple de màquina d'estats a Unity

Si es compleix una condició definida entre *Any State* i un altre estat, es produeix la transició sense importar a quin estat es trobi la màquina i si els estats estan connectats entre si.

Pel que fa a *Entry* i *Exit,* estan pensats per implementar màquines d'estat que incorporin sub-màquines com a estat. D'aquesta manera, es poden definir les transicions que cal aplicar al entrar en la sub-màquina i les que fan que es surti d'ella i es retorni a la màquina principal. Degut a que les màquines d'estat necessàries per la implementació del projecte no inclouen sub-màquines, l'estat *Entry* es connecta a l'estat per defecte i l'estat *Exit* no és accessible.

El procediment depèn en gran mesura dels estats que es requereixin, i serà detallat en apartats posteriors quan es defineixi el procés d'implementació per cada un dels personatges i objectes.

5.3.2. Nivells

Per a la construcció dels nivells s'ha decidit treballar amb *tiles*, utilitzant l'editor de mapes *Tiled*, que és un software gratuït que facilita el disseny de mapes basats en *tiles*, i *Tiled2Unity*, una eina que permet importar a *Unity* els nivells creats amb *Tiled*.

En un videojoc basat en *tiles*, la zona de joc està constituïda per petits rectangles, hexàgons o quadrats que contenen imatges, que en anglès s'anomenen *tiles.* Aquests *tiles* es disposen en forma de graella, formant una imatge continua que s'utilitza per crear el nivell. En el cas del projecte s'ha treballat amb *tiles* quadrats de 16x16 píxels.

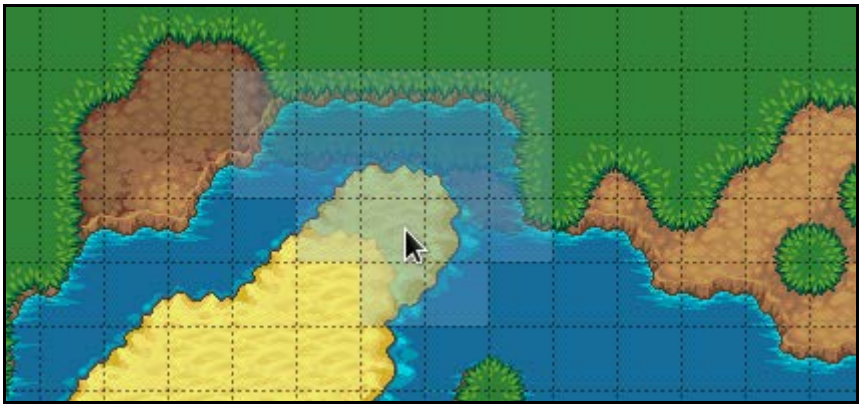

Figura 5.18. Exemple de mapa basat en tiles

Habitualment es disposen tots els *tiles* a utilitzar en la creació del nivell o nivells en una única imatge anomenada *tileset*, on s'inclouen tant els elements de terreny com els paisatges que formaran el fons.

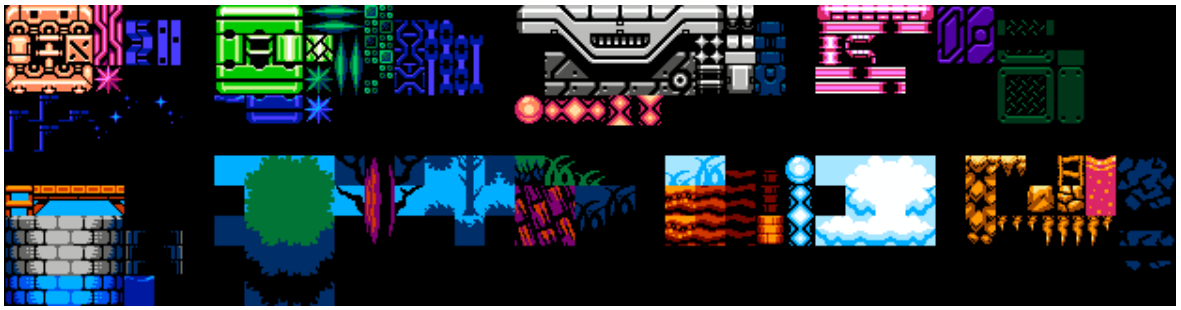

Figura 5.19. Exemple de tileset utilitzat al prototip

En el disseny dels nivells del prototip, tal com s'ha comentat amb anterioritat, s'ha utilitzat l'editor de mapes *Tiled* per generar els nivells i la eina *Tiled2Unity* per importar aquestes imatges a *Unity*.

En primer lloc, cal importar el *tileset* a utilitzar a *Tiled*, així com crear una graella de les

dimensions desitjades. Es crea el nivell disposant els diferents *tiles* de tal manera que formin una imatge coherent. *Tiled* permet treballar amb capes per facilitar el disseny.

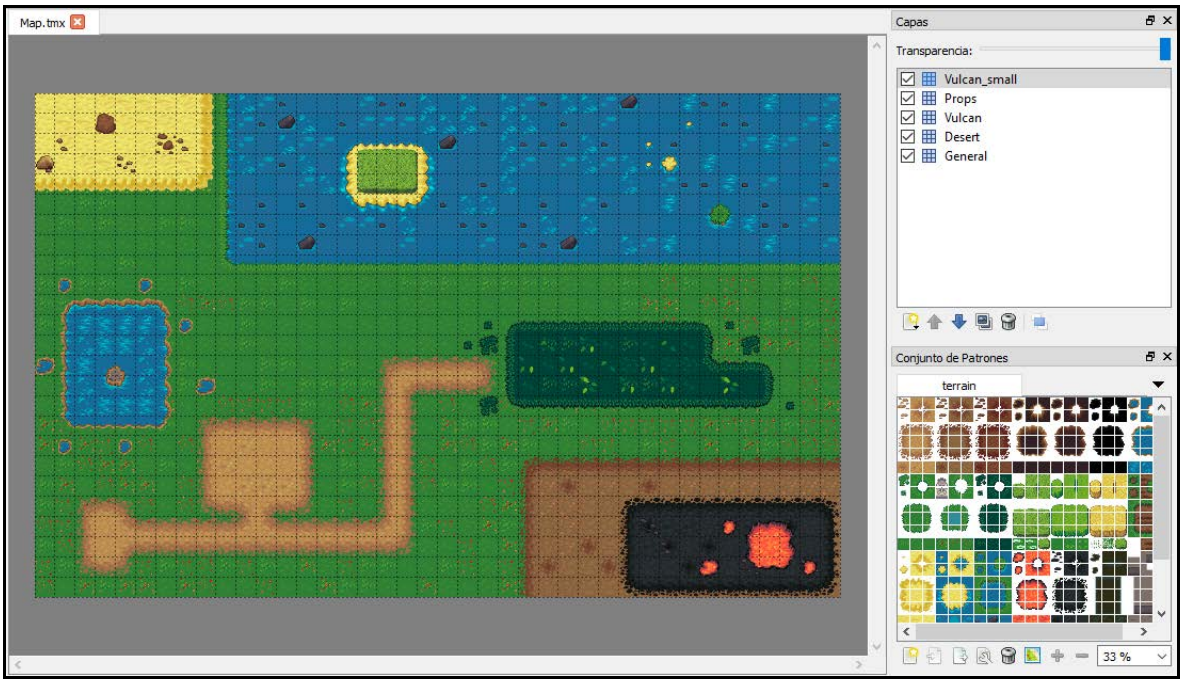

Figura 5.20. Mapa del món, creat amb Tiled

Un cop s'ha dissenyat el nivell, cal importar-lo utilitzant *Tiled2Unity*. Aquesta eina posa a disposició de l'usuari una sèrie de *scripts* que transformen cada una de les capes del mapa a *Tiled* en un *prefab* de *Unity*. Un cop es tenen els terrenys que formen el nivell a *Unity*, només cal definir els col·lisionadors que conformen el terra, que s'utilitzaran quan es calculin les físiques dins del motor de *Unity*.

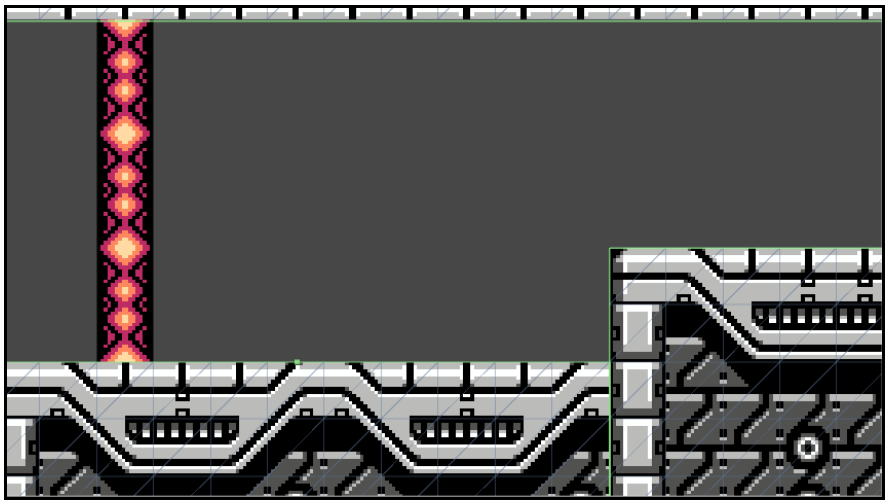

Figura 5.21. Terreny importat a Unity, amb col·lisionadors (línies verdes)

6. Implementació del personatge principal

En aquest apartat es detalla la implementació de les característiques descrites a la secció Personatges jugables, que es troba a la pàgina 12 d'aquesta memòria. Per realitzar totes les accions de que disposa el personatge, s'ha utilitzat la classe *PlayerController*, que es troba al script *PlayerController.cs*.

6.1. Màquina d'estats

Per controlar les animacions del personatge, s'utilitza el sistema de màquines d'estat incorporat a Unity, anomenat Mechanim. La metodologia es basa en definir cada animació com a un estat, i imposar les condicions que s'han de donar per a que el controlador d'animacions realitzi la transició entre els diferents estats. A la **Figura 6.1** es mostra l'estructura de la màquina d'estats aplicada al personatge principal.

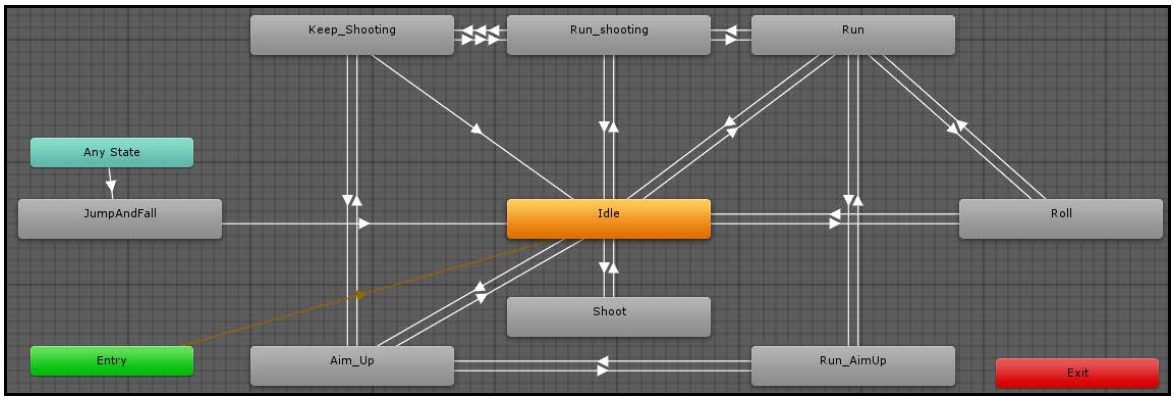

Figura 6.1. Màquina d'estats del controlador d'animacions del protagonista

El llistat d'estats de que es disposa és el següent:

- ***Idle*:** És l'estat per defecte. L'animació associada a aquest estat és simplement el personatge parat.
- ***Shoot:*** Animació de dispar. S'utilitza únicament pel primer dispar, si el jugador continua pressionant el botó de dispar es realitza una transició a *Keep Shooting* via script.
- ***Roll:*** Animació del personatge rodant.
- ***Run:*** Animació del personatge corrent.

- ***Run Shooting:*** Animació del personatge corrent i disparant a la vegada.
- ***Keep Shooting:*** Aquesta animació només es composa d'un fotograma, el personatge amb el braç estirat, el fotograma final de l'animació *Shoot*. Es realitza la transició a aquest estat si el jugador continua disparant després del primer dispar.
- ***Aim Up:*** El personatge apunta amunt.
- ***Run Aim Up:*** El personatge apunta amunt i corre al mateix temps.
- ***Jump And Fall:*** Es reprodueix un fotograma determinat en funció de la velocitat vertical del jugador, produint tant l'animació de salt com la de caiguda lliure.

Per realitzar la transició entre aquests estats, es defineixen els següents paràmetres al controlador d'animacions, que es modifiquen de la manera requerida als *scripts* que controlen el jugador:

Variable	Tipus	Valor
Speed	Real	Velocitat horitzontal del personatge
vSpeed	Real	Velocitat vertical del personatge
Grounded	Booleà	Vertader si el personatge està en contacte amb el terra, fals en cas contrari.
Shooting	Booleà	Vertader si el botó de dispar està pressionat, fals en cas contrari.
AimingUp	Booleà	Vertader si el jugador apunta cap amunt, fals en cas contrari
Rolling	Booleà	Vertader si el jugador es troba en l'animació de rodar, fals en cas contrari
Laser	Booleà	Vertader si l'arma secundària làser es troba en ús, fals en cas contrari

Com s'ha explicat amb anterioritat, l'estat Entry es connecta a l'estat per defecte per tal de que al iniciar l'escena es reprodueixi l'animació per defecte. Les condicions per les transicions sempre són del tipus booleà, i es poden definir en funció de més d'un paràmetre, treballant entre si com a producte de booleans (operador AND). Hi ha transicions que es mostren amb més d'un punter, això és degut a que presenten més d'un grup de condicions, actuant entre ells com a suma booleana (operador booleà OR).

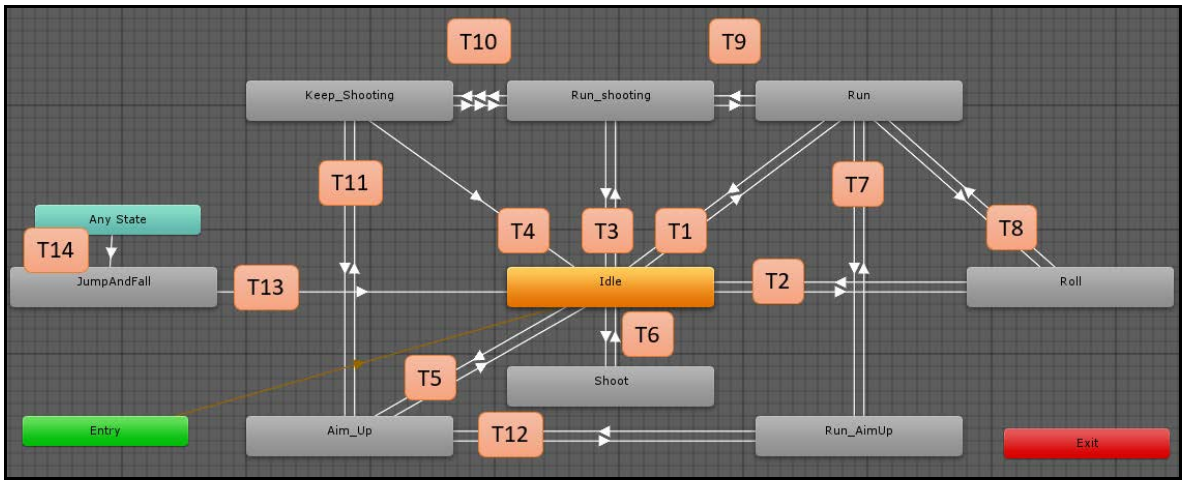

Figura 6.2. Màquina d'estats amb transicions numerades

El llistat de transicions i les condicions que s'han de donar per que es produeixin es detalla a continuació, en base a la numeració mostrada a la ***Figura 6.2***. El valor de la variable *Speed* per al qual es produeix la transició és arbitrari, s'ha definit mitjançant prova i error.

	Estats	*Transicions*	*Condicions*
T1	*Idle ⇌ Run*	*Idle → Run*	*Speed > 0.1*
		Run → Idle	*Speed < 0.1*
T2	*Idle ⇌ Roll*	*Idle → Roll*	*Rolling = Vertader*
		Roll → Idle	*Rolling = Fals*
T3	*Idle ⇌ Run Shooting*	*Idle → Run Shooting*	*Speed > 0.1 & Shooting = Vertader*
		Run Shooting → Idle	*Speed < 0.1 & Shooting = Fals & Laser = Fals*
T4	*Keep Shooting → Idle*	*Keep Shooting → Idle*	*Shooting = Fals & Laser = Fals*
T5	*Idle ⇌ Aim Up*	*Idle → Aim Up*	*AimingUp = Vertader*
		Aim Up → Idle	*AimingUp = Fals*

T6	*Idle ⇌ Shoot*	*Idle → Shoot*	*Shoot = Vertader*
		Shoot → Idle	*Shoot = Fals*
T7	*Run ⇌ Run Aim Up*	*Run → Run Aim Up*	*AimingUp = Vertader*
		Run Aim Up → Run	*AimingUp = Fals*
T8	*Run ⇌ Roll*	*Run → Roll*	*Rolling = Vertader*
		Roll → Run	*Speed > 0.1 & Rolling = Fals*
T9	*Run ⇌ Run Shooting*	*Run → Run Shooting*	*Shooting = Vertader*
		Run Shooting → Run	*Speed > 0.1 & Shooting = Fals & Laser = Fals*
T10	*Run Shooting ⇌ Keep Shoting*	*Run Shooting → Keep Shooting*	*{Speed < 0.1 & Shooting = Vertader} O {Speed < 0.1 & Laser = Vertader}*
		Keep Shooting → Run Shooting	*{Speed > 0.1 & Shooting = Vertader} O {Speed > 0.1 & Laser = Vertader}*
T11	*Keep Shoting ⇌ Aim Up*	*Keep Shooting → Aim Up*	*Aiming Up = Vertader & Laser = Fals*
		Aim Up → Keep Shooting	*Aiming Up = Fals & Shooting = Fals*
T12	*Aim Up ⇌ Run Aim Up*	*Aim Up → Run Aim Up*	*Speed > 0.1*
		Run Aim Up → Aim Up	*Speed < 0.1*

T13	*Jump And Fall → Idle*	*Jump And Fall → Idle*	*Grounded = Vertader*
T14	*Any State → Jump And Fall*	*Any State → Jump And Fall*	*Grounded = Fals*

6.2. Moviments

En aquesta secció es descriuen els mètodes emprats per aconseguir els moviments definits amb anterioritat. En general els moviments han sigut implementats utilitzant les funcionalitats del component *Rigidbody2D*. Les accions que s'han de dur a terme s'han inclòs al script *PlayerController.cs*.

6.2.1. Córrer

Figura 6.3. Personatge corrent

Córrer és el moviment principal del personatge, ja que és el que el permet moure's per sobre de les plataformes. La idea per realitzar aquest moviment és utilitzar l'eix d'entrada horitzontal definit a *Input*, que retorna un nombre real entre -1.0 i 1.0 en funció de la intensitat amb que es pressiona l'eix, i la propietat *velocity* de la classe *Rigidbody2D*.

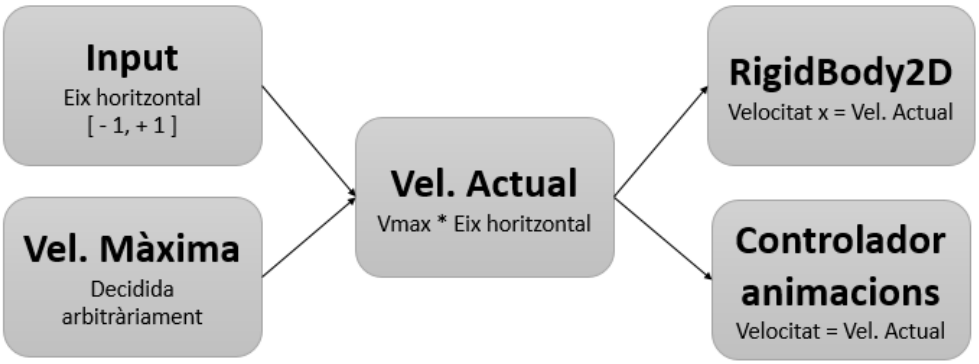

Figura 6.4. Implementació del desplaçament

D'aquesta manera, es defineix la velocitat en X del sòlid rígid com el valor d'entrada de l'eix horitzontal multiplicat per la velocitat màxima del personatge. Un cop es té el valor de la velocitat, s'aplica també al controlador d'animacions per tal que es reprodueixi l'animació corresponent. Totes aquestes accions es realitzen a la funció *FixedUpdate*, que s'avalua a cada fotograma fixat pel motor de físiques.

6.2.2. Saltar

Figura 6.5. Personatge saltant

Saltar també és un moviment important, ja que possibilita al protagonista superar diferències de nivell o espais buits entre les plataformes. Per implementar aquest moviment, de nou es recorre al sòlid rígid associat al personatge, en aquest cas utilitzant la funció *AddForce,* que donat un vector de força, l'aplica sobre el sòlid rígid. També es té en compte el botó de salt definit a *Input*, per determinar en quin moment cal aplicar la força. Com a variable auxiliar s'ha definit un booleà que indica si el jugador esta en contacte amb el terra o no, per tal d'impossibilitar la realització de més d'un salt consecutiu sense tocar el terra.

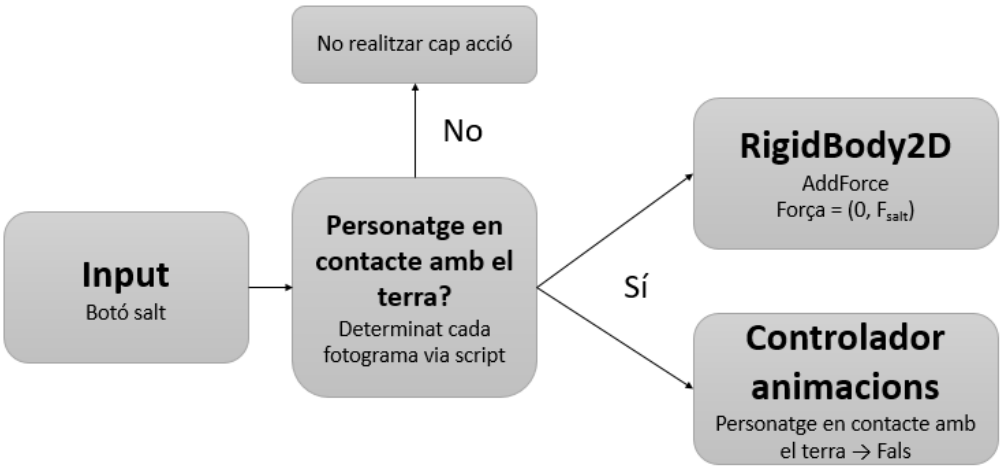

Figura 6.6. Implementació del salt

Si es donen les condicions requerides, que són que a la mateixa vegada el personatge

estigui en contacte amb el terra i el botó de salt sigui pressionat, s'aplica una força amb component horitzontal nul·la que fa que el personatge es desplaci verticalment, aconseguint així l'efecte de salt. Les accions es realitzen a la funció *Update*, que s'avalua a cada fotograma.

6.2.3. Rodar

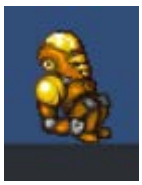

Figura 6.7. Personatge rodant

Aquest moviment fa que el personatge sigui immune durant el temps en que es reprodueix l'animació, permetent així que el jugador pugui esquivar els projectils enemics. La implementació d'aquest moviment es basa en utilitzar un booleà per determinar si el jugador és immune al dany, que per defecte és fals i es defineix com a vertader quan el jugador prem el botó de rodar, sempre que el personatge estigui en contacte amb el terra. També s'afegeix una petita força al sòlid rígid associat al personatge amb la funció *AddForce* per aconseguir un efecte de tombarella. Totes aquestes accions es troben dins de la funció *Update*, que s'executa cada fotograma.

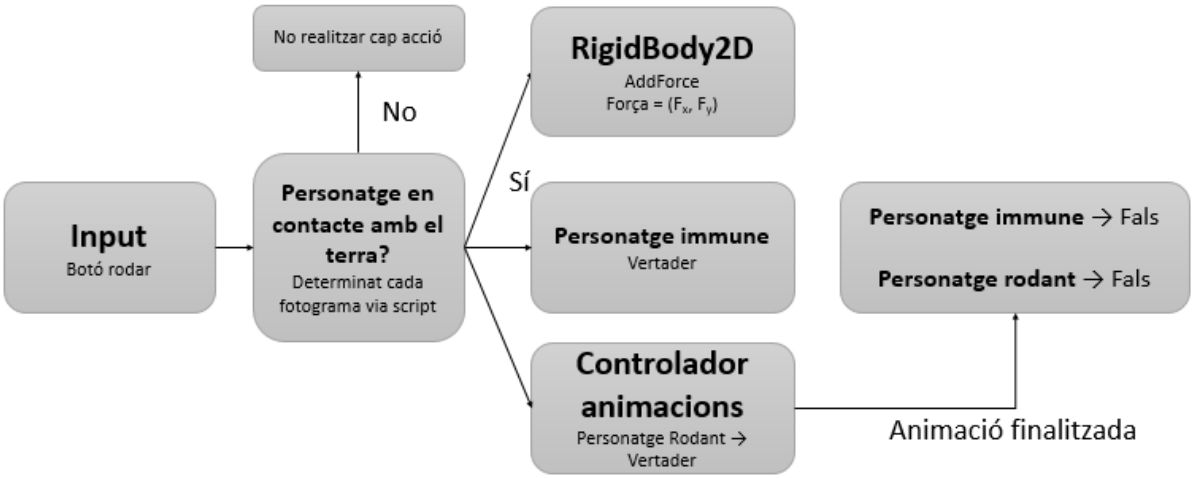

Figura 6.8. Implementació del moviment Rodar

En l'últim fotograma de l'animació de rodar, s'executa una funció que torna a definir el booleà d'immunitat com a fals, permetent que el personatge pugui sofrir danys un altre cop. També defineix com a fals el booleà del personatge rodant al controlador d'animacions, per tal de

realitzar la transició cap a un altre estat.

6.3. Armes

Per la implementació de les armes s'han utilitzat generalment *prefabs* de cada tipus de projectil creant una nova instància quan sigui necessari, la posició inicial de la qual és determina a partir d'un *GameObject* buit i fill del *GameObject* del personatge, que es situa al punt desitjat. També s'incorporen col·lisionadors per detectar el xoc entre el projectil i els altres elements de l'escena.

Pel que fa als scripts que controlen les armes, cada projectil té adjunt un *script* que controla el seu comportament i el dany que causa, mentre que la creació de la instància es duu a terme al script *PlayerController.cs*. En les armes secundàries, que disposen d'un sistema de munició, els valors de munició disponible es guarda al script *PlayerStats.cs.* A continuació es detalla la implementació de cadascuna de les armes.

6.3.1. Canó làser, mode foc ràpid

Figura 6.9. Personatge disparant en horitzontal i vertical

El canó làser en mode foc ràpid és l'arma principal del protagonista, i és la més utilitzada durant el transcurs del joc. El procés d'implementació es pot dividir en dos parts, la creació de la instància del *prefab* amb el raig làser i el comportament del propi projectil.

Pel que fa a la creació de la instància, es controla a través del personatge, al *script PlayerController.cs.* En primer lloc s'utilitza la classe *Input* per detectar si el botó de dispar ha estat premut i en cas afirmatiu es crea una instància de la bala a la mateixa vegada que es reprodueix l'animació corresponent. Un cop ha començat la seqüencia de dispar, les bales només es generen cada un cert temps definit de manera arbitrària, fins que el jugador deixa de polsar el botó. Per aconseguir aquest comportament, s'utilitza una variable a mode de comptador, que torna a 0 cada cop que es genera una bala. Degut a que es pot disparar verticalment, cal utilitzar un booleà, que és vertader quan el jugador està apuntant amunt, ja que la posició en la que cal crear la bala és diferent en cadascun dels casos.

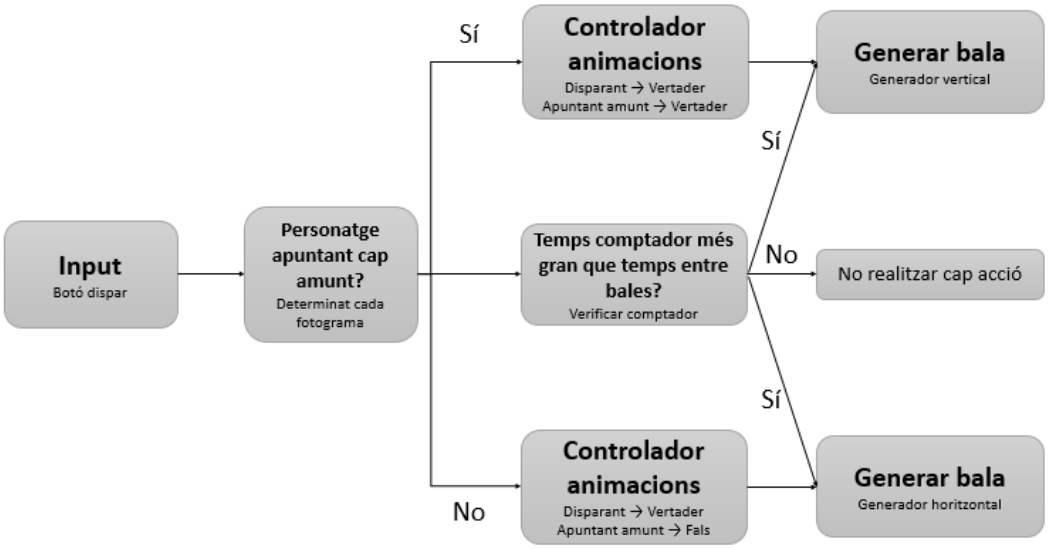

Figura 6.10. Implementació del canó làser, part controlada pel personatge

Per generar les bales, s'utilitza un *GameObject* buit que és fill del GameObject del personatge, i que té adjunt un *script* anomenat *BalaSpawn.cs*. Degut a aquesta relació pare / fill, el generador es manté sempre al mateix punt en relació al personatge. L'algorisme per generar bales és senzill, dins del *script* es defineix la funció *Spawn* que, donat un *prefab* de bala que conté informació sobre si serà disparada en horitzontal o vertical, comprova la direcció i crea una instància de la bala a la posició on es troba el generador, rotant-la 90º si la bala és vertical. L'únic que cal fer és executar aquesta funció en cas que es vulgui generar una bala al diagrama de la **Figura 6.10**.

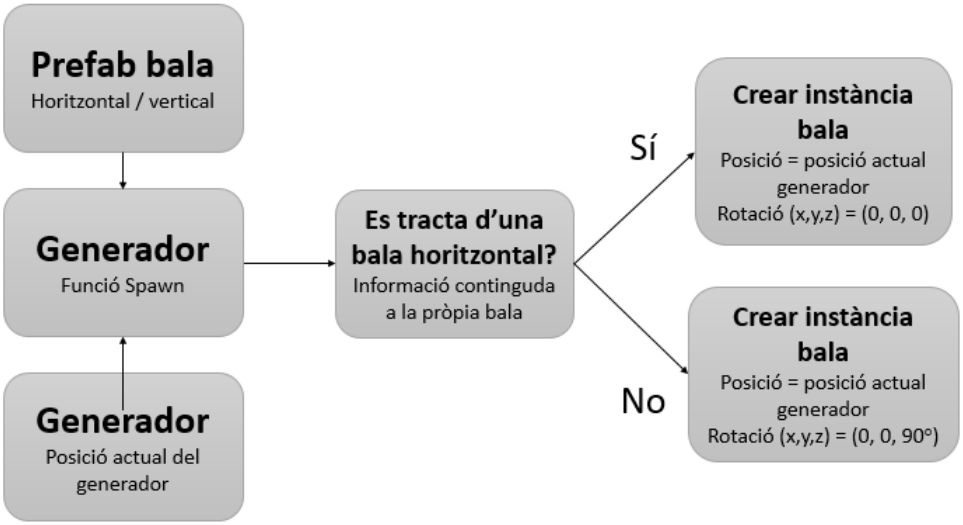

Figura 6.11. Funcionament del generador de bales

Un cop generada la instància de bala, el propi projectil té adjunt un script que determina el seu comportament. Aquest script s'anomena *Bullet.cs,* i s'utilitzen les propietats de sòlid rígid proporcionades per la classe *RigidBody2D* per aplicar una força puntual en el moment que és generada mitjançant la funció *AddForce*. Un cop generada, en primer lloc es comprova si la bala és horitzontal o vertical (característica definida prèviament), i en funció del tipus de bala s'aplica una força diferent. Per evitar els efectes de la gravetat en el motor de físiques i aconseguir un moviment rectilini, les bales horitzontals tenen bloquejat el desplaçament en l'eix Y i les bales verticals tenen bloquejat el desplaçament en l'eix X.

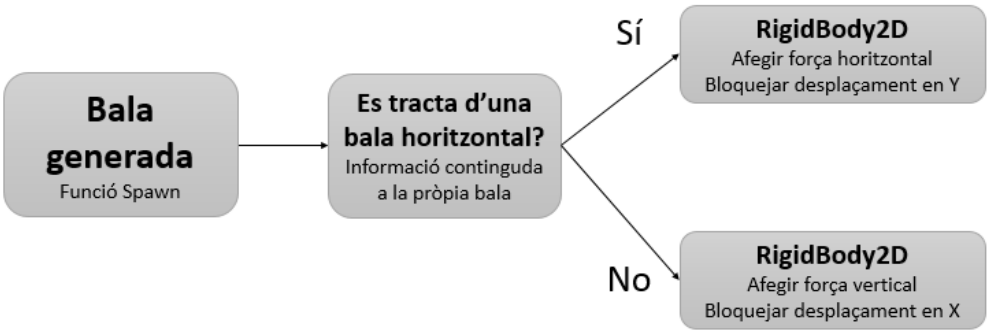

Figura 6.12. Primer instant d'una bala generada

Quan la bala ja disposa de moviment, mitjançant la funció *OnTriggerEnter2D* de la classe *Collider2D*, es detecten les col·lisions de la bala amb altres objectes. Si la bala col·lideix amb un objecte qualsevol, simplement es destrueix el *GameObject* de la bala, però si l'objecte amb el que col·lideix té la etiqueta *Enemy*, utilitzada a tots els enemics del joc, causa una quantitat de dany a l'enemic definida per un nombre real.

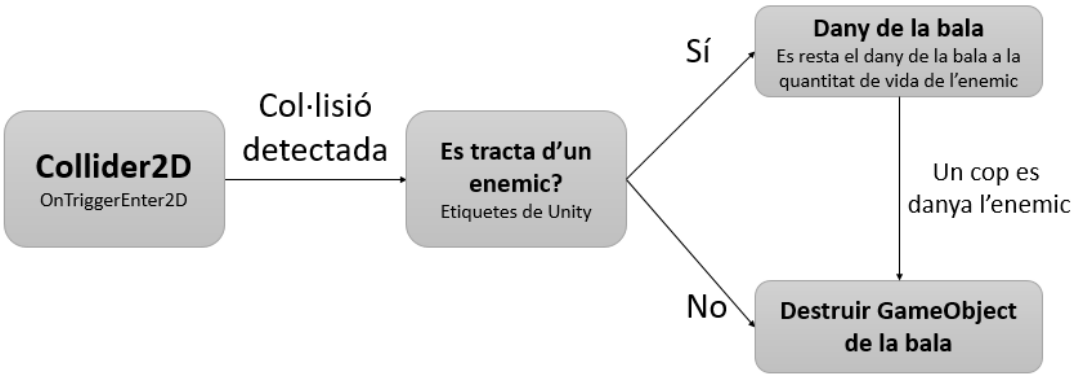

Figura 6.13. Comportament de la bala en front de col·lisions

6.3.2. Implementació general d'armes secundàries

El procediment d'implementació de les armes secundàries és molt similar en tots els casos, variant únicament el comportament del projectil. Per aquest motiu en aquest apartat es descriu la metodologia general d'implementació d'armes secundàries, així com la gestió de la seva munició. El comportament de cada tipus d'arma secundària és tractat de manera específica al seu apartat.

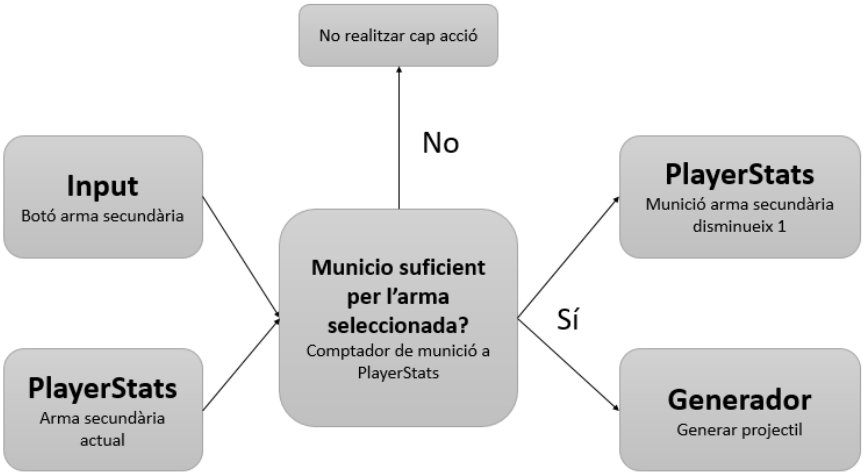

Figura 6.14. Implementació de les armes secundàries

Per gestionar les armes secundàries, s'utilitza la classe *PlayerStats*, inclosa al script *PlayerStats.cs*, que permet determinar quina és l'arma secundària actual, així com la seva munició. El projectil es genera únicament si la munició és superior a 0, i aquesta es disminueix en 1 cada cop que s'utilitza l'arma.

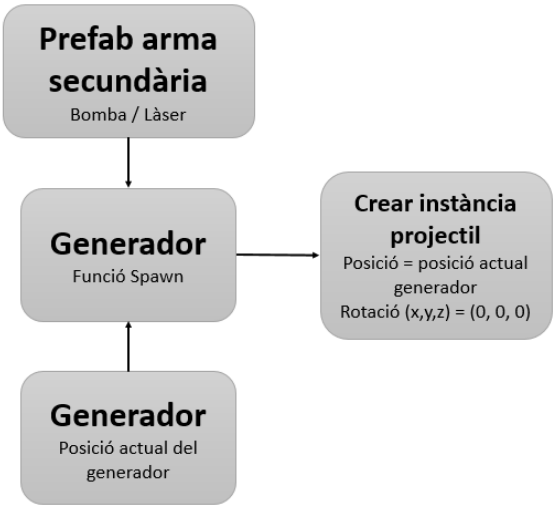

Figura 6.15. Funcionament generador d'armes secundàries

El procediment per generar una instància del projectil es més simple que la del canó làser en mode foc ràpid, ja que degut a que es llancen únicament en horitzontal, no cal considerar si el projectil és horitzontal o vertical quan es crea l'objecte.

6.3.3. Bomba

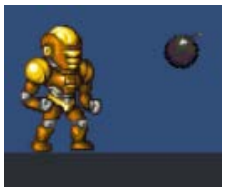

Figura 6.16. Personatge llançant una bomba

La principal arma secundària del protagonista és la bomba, que ha estat implementada utilitzant la funció *OverlapCircleAll* inclosa a la classe *Physics2D*, que retorna una llista dels col·lisionadors que es troben dins de l'àrea d'un cercle definit per la seva posició i el seu radi.

Per tal d'aconseguir un millor aspecte visual en el comportament de les bombes, s'ha inclòs un component *RigidBody2D* que tracta la bomba com un sòlid rígid, així com un material de físiques, que s'afegeix al col·lisionador de l'objecte.

Els materials de físiques a *Unity* modifiquen el comportament de les col·lisions, permetent definir el coeficient de fregament i el nivell de rebot, que té un rang de valors de 0 a 1, on 0 és una col·lisió sense rebot i 1 un rebot perfecte sense pèrdua d'energia.

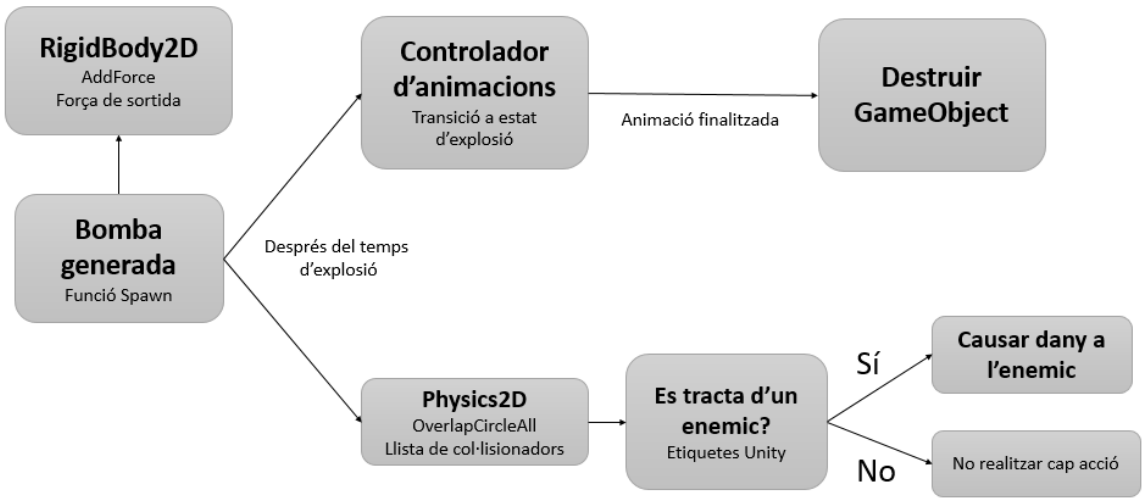

Figura 6.17. Comportament de les bombes

En el moment en que la bomba és instanciada, se li aplica una forca utilitzant la funció

AddForce de la classe *RigidBody2D*, i es dóna ordre d'executar la funció *Explota* quan passi el temps d'explosió, que es defineix arbitràriament. La funció *Explota* realitza una transició en el controlador d'animacions de la bomba cap a l'estat d'explosió, i executa la funció *OverlapCircleAll* de la classe *Physics2D*, que retorna una llista dels components *Collider2D* que es troben dins del radi de la explosió com s'ha explicat anteriorment en aquest apartat. D'aquesta manera, només cal saber si els col·lisionadors de la llista estan continguts en un *GameObject* etiquetat com a enemic, i produir la quantitat de dany corresponent en cas afirmatiu. Finalment, quan l'animació d'explosió finalitza, es destrueix el *GameObject* de la bomba.

6.3.4. Raig làser

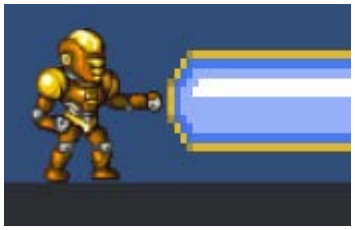

Figura 6.18. Personatge disparant un raig làser

El raig làser té un ús més limitat que les bombes degut al seu gran poder. Per implementar-lo s'ha utilitzat bàsicament la funció *OnTriggerStay2D* de la classe *Collider2D*, així com un booleà definit a la classe *PlayerController*, que determina si el làser està sent disparat o no i que per defecte és fals. Quan s'utilitza el làser no es pot realitzar cap altre acció fins que la durada del làser hagi acabat, excepte saltar.

En primer lloc, quan es rep la informació de que el botó d'arma secundària ha estat premut, es defineix el booleà del làser com a vertader, es genera la instància del raig làser i es realitza la transició a l'animació de dispar al controlador d'animacions. Quan l'animació acaba, es destrueix el *GameObject* del raig i es torna a definir el booleà del làser com a fals, per tornar a l'estat corresponent al animador i permetre que el personatge realitzi accions un altre cop.

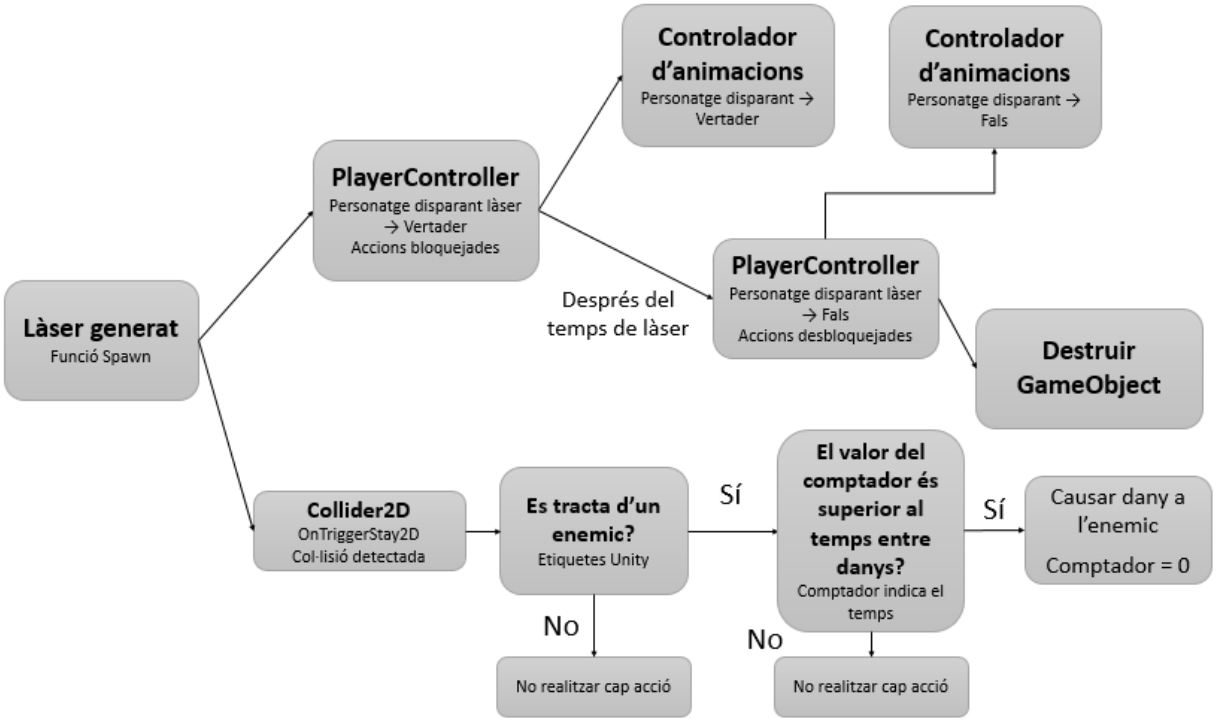

Figura 6.19. Implementació del raig làser

Pel que fa al comportament del projectil del làser, s'utilitza un comptador per causar danys cada un cert temps, utilitzant la funció *OnTriggerStay2D*, que s'executa cada fotograma consecutiu en que es mantingui la col·lisió. Cada vegada que es causa dany, el comptador torna a 0.

En el cas del làser, es requereix que el projectil instanciat sigui fill del generador per aconseguir que el raig estigui en tot moment a la posició desitjada, la mà del personatge.

7. Implementació d'enemics, objectes i nivells

En aquest apartat es detalla la forma en que han sigut implementats tots els enemics, objectes i nivells que s'inclouen al prototip. Les interaccions d'aquests elements amb el personatge principal es duen a terme, generalment, modificant els valors de les variables contingudes a la classe *PlayerStats* adjunta al protagonista, que es defineix al *script PlayerStats.cs*.

7.1. Enemics

S'han implementat dos tipus d'enemics dels que es plantegen a l'apartat de disseny (pàgina 16), concretament els del tipus *Patrulla amb dispar* i *Fix amb dispar*. En tots dos casos, el seu comportament varia quan veuen al protagonista, per aquest motiu s'explica l'algoritme general d'implementació.

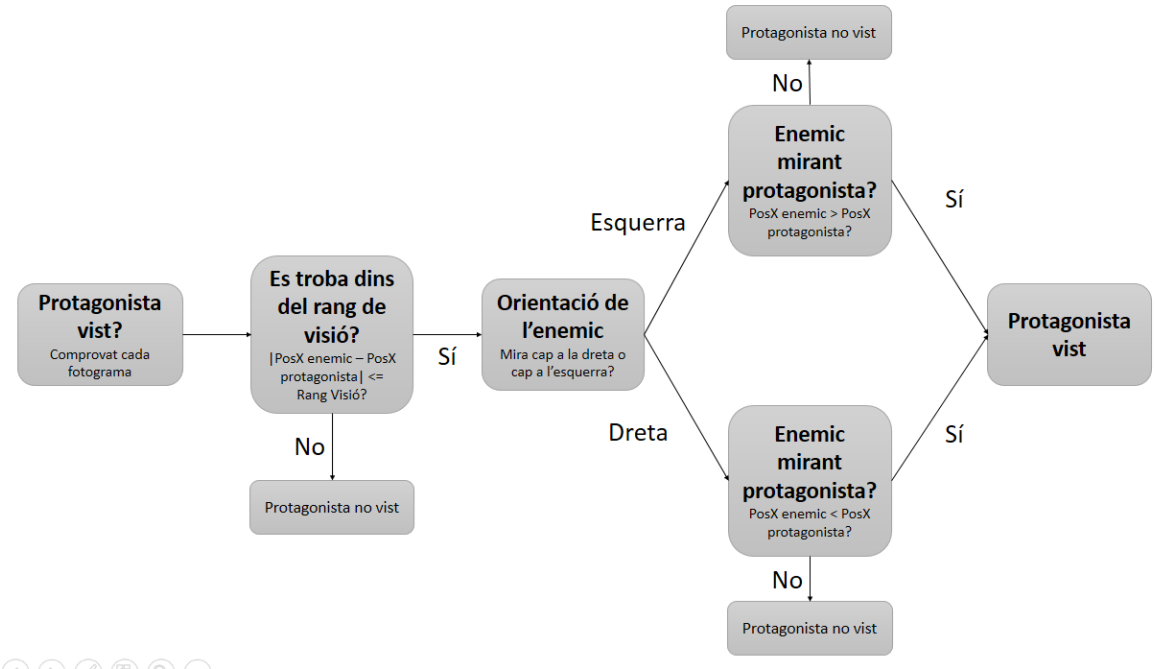

Figura 7.1. Algorisme utilitzat per determinar si l'enemic veu al protagonista

Per determinar si l'enemic veu al protagonista, en primer lloc es comprova si el valor absolut de la diferència de posicions és menor o igual al rang de visió que té l'enemic. En cas afirmatiu, es comprova l'orientació de l'enemic mitjançant un booleà que és vertader si

l'enemic mira cap a la dreta. Un cop coneguda l'orientació, cal saber en quin costat es troba el personatge. Si la posició del jugador en l'eix X és superior a la de l'enemic, es troba a la dreta d'aquest, en cas contrari es troba a l'esquerra. Amb aquestes dades es pot determinar si l'enemic veu al personatge.

En els apartats següents es descriu com s'ha aconseguit el comportament dels diferents tipus d'enemics. Només incorporen una animació, per tant no es descriurà en detall la màquina d'estats com si s'ha fet amb el protagonista.

7.1.1. Drac (Patrulla amb envestida)

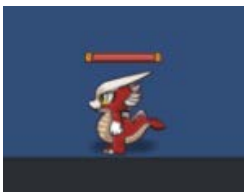

Figura 7.2. Enemic drac

El drac és un enèmic que realitza la funció de patrulla amb envestida. Com s'indica a l'apartat de disseny, la seva tasca consisteix en desplaçar-se de forma continua dins d'un espai definit, i un cop pot veure al protagonista, comença a perseguir-lo augmentant la seva velocitat. Per implementar aquestes característiques s'ha fet us de la classe *RigidBody2D* per dotar a l'enemic de moviment i s'ha determinat si el protagonista es vist pel drac en funció de les seves posicions i l'orientació del drac, que ve determinada per un booleà que és vertader quan el drac mira cap a la dreta.

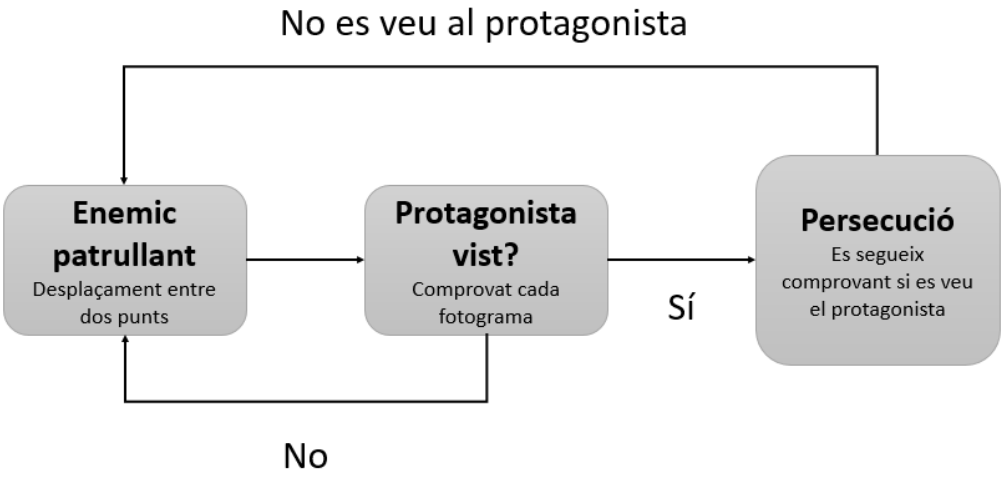

Figura 7.3. Comportament d'un enemic del tipus patrulla amb envestida

El comportament de l'enemic es basa en desplaçar-se entre dos punts, comprovant sempre si pot veure al protagonista tenint en compte el seu rang de visió. Quan es doni el cas de que el drac vegi al protagonista, comença la fase de persecució, que termina quan el personatge principal s'allunya una distància suficient com per estar fora del rang de visió del drac. Durant aquesta fase de persecució l'enemic augmenta la seva velocitat.

7.1.2. Torreta (Fix amb dispar)

Figura 7.4. Enemic torreta

L'altre enemic implementat és la torreta. Aquest enemic no es mou, i dispara projectils quan veu al jugador. Per la seva implementació s'ha seguit un procediment molt similar al de les armes del protagonista.

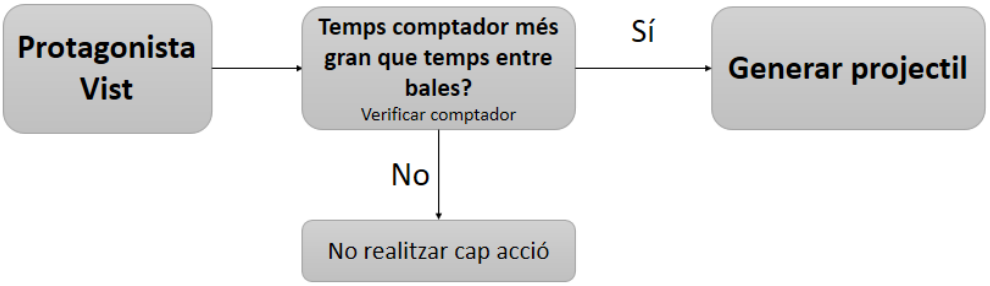

Figura 7.5. Comportament dels enemics un cop veuen al protagonista

Les condicions que s'han de donar per generar un projectil són molt similars a les del canó làser en mode foc ràpid del protagonista (pàgina 55), però així com en el cas del protagonista es requeria pressionar el botó de dispar, en el cas dels enemics es generen projectils quan aquests veuen al protagonista.

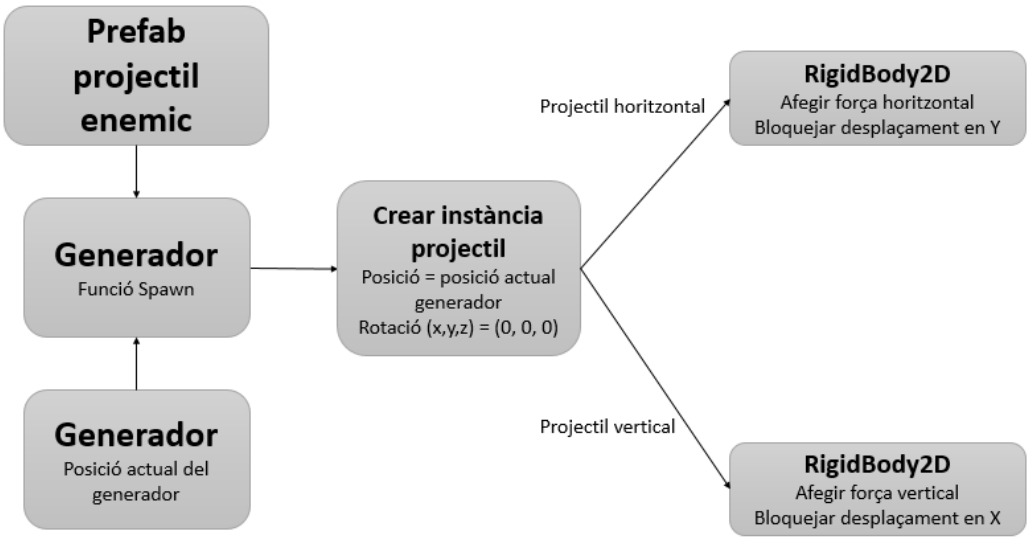

Figura 7.6. Generació de projectils enemics

La generació de projectils també s'assimila a la implementació de les bales del protagonista, utilitzant un generador que crea una instància del projectil en la seva posició. En funció de si l'enemic dispara de forma vertical o horitzontal, s'aplica una força diferent sobre cada instància del projectil utilitzant la funció *AddForce* de la classe *Rigidbody2D,* a l'hora que es bloqueja el desplaçament en l'eix X o Y del projectil segons convingui.

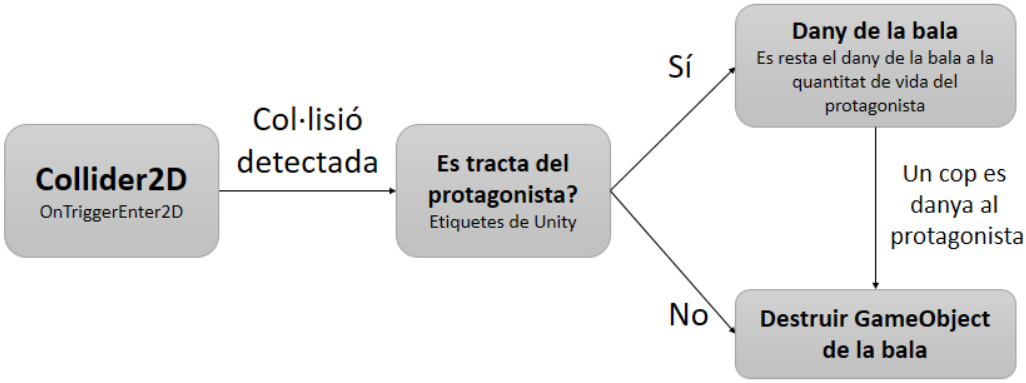

Figura 7.7. Comportament dels projectils enemics

Pel que fa al comportament dels projectils generats, també és molt similar al dels projectils del canó làser en mode foc ràpid, però amb la particularitat que cal que els projectils causin dany al personatge principal. Com en el cas de les bales del protagonista, quan es detecta el contacte, es comprova si el *GameObject* que conte el col·lisionador té la etiqueta *Player*, que únicament està assignada al protagonista. En cas afirmatiu, es resta el dany del projectil a la

vida del protagonista, continguda al script *PlayerStats.cs*.

Figura 7.8. Mini torreta

Un cas particular de torreta és la mini-torreta, que es situa a la part superior de les plataformes i dispara contra el terra. En el cas d'aquest enemic, s'utilitza a mode de trampa, doncs dispara de forma contínua sense necessitat de veure al protagonista.

7.2. Objectes

Tal com es va mostrar a la fase de disseny, els objectes es poden classificar en tres tipus: col·leccionables, usables i interactius. Tots els objectes de cada tipus es comporten de manera molt similar, pel que en els apartats següents es descriu la metodologia general d'implementació per cada tipus d'objecte, així com una llista dels objectes disponibles al prototip.

7.2.1. Objectes col·leccionables

Els objectes col·leccionables són aquells que el jugador pot acumular, amb un propòsit diferent en cada cas. A continuació es mostra la llista dels objectes col·leccionables inclosos al prototip, així com la seva representació dins del joc.

Objecte	Funció	*Sprite*	Quantitat
Monedes	Incrementa el comptador de monedes segons el valor de la moneda.		1
			5
			15
Munició Bombes	Incrementa la munició de bombes.		3

Munició Làser	Incrementa la munició de làser.		1
Poció de vida	Incrementa la quantitat de pocions de vida que té el personatge.		1

Pel que fa a la implementació, s'utilitza la funció *OnTriggerEnter2D* de la classe *Collider2D*. Mitjançant aquesta funció es detecten totes les col·lisions amb l'objecte, i es comprova si el *GameObject* que ha entrat amb contacte amb l'objecte col·leccionable és el jugador mitjançant les etiquetes de *Unity*. En cas afirmatiu, s'incrementa la quantitat del comptador associat a l'objecte i es destrueix el *GameObject*, desapareixent així de l'escena.

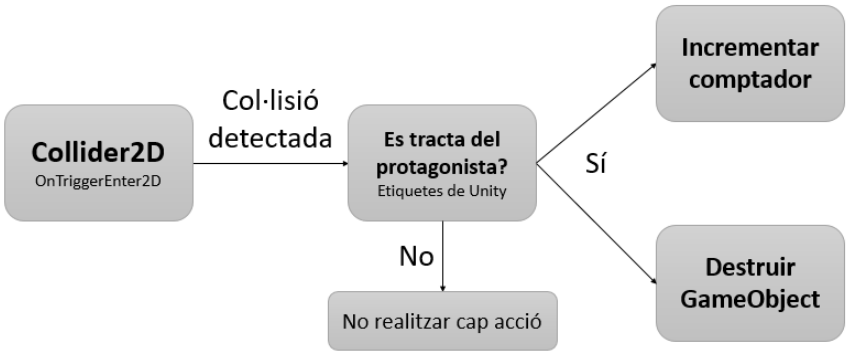

Figura 7.9. Implementació dels objectes col·leccionables

7.2.2. Objectes consumibles

Els únics objectes consumibles inclosos al prototip són les pocions de vida. Per la seva implementació s'utilitza la classe *Input* per detectar si el botó d'objectes consumibles és polsat, i sempre que hi hagi pocions disponibles, es modifica el valor de la vida actual del personatge, augmentant-lo en un 30% de la vida total. Addicionalment, es disminueix la quantitat de pocions disponibles en 1. Tant la vida del personatge com la quantitat de pocions disponibles es guarden al script *PlayerStats.cs*, adjunt al *GameObject* del protagonista.

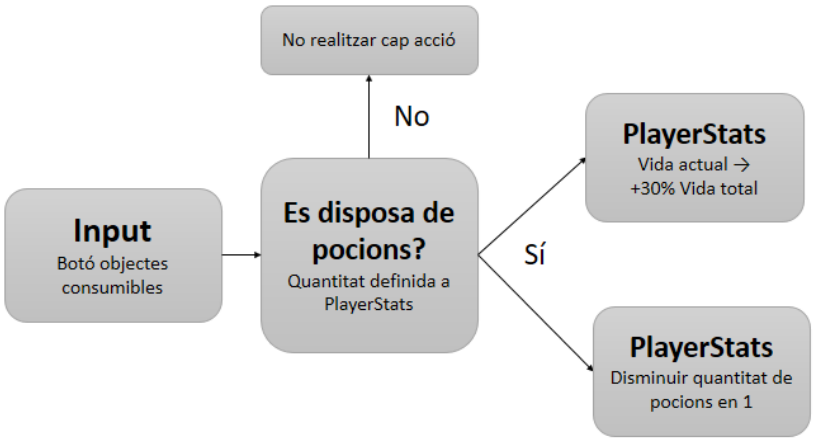

Figura 7.10. Implementació de les pocions de vida

7.2.3. Objectes interactius

En quant a objectes interactius, l'únic objecte disponible és l'escala. La seva implementació es basa en les funcions *OnTriggerEnter2D, OnTriggerStay2D i OnTriggerExit2D*, totes elles pertanyents a la classe *Collider2D*, realitzant les accions oportunes sempre que l'objecte que col·lideixi amb les escales sigui el personatge principal. Com a variable auxiliar es defineixen un booleà que és vertader si el jugador està en contacte amb l'escala i que per defecte és fals. Es desitja que si el jugador no pressiona cap botó mentre es troba pujant l'escala, el personatge es quedi a la mateixa posició, per tant quan s'entra en contacte amb l'escala es fixa la gravetat del component Rigidbody2D del personatge a 0, i es restableix al seu valor original un cop s'acaba la col·lisió.

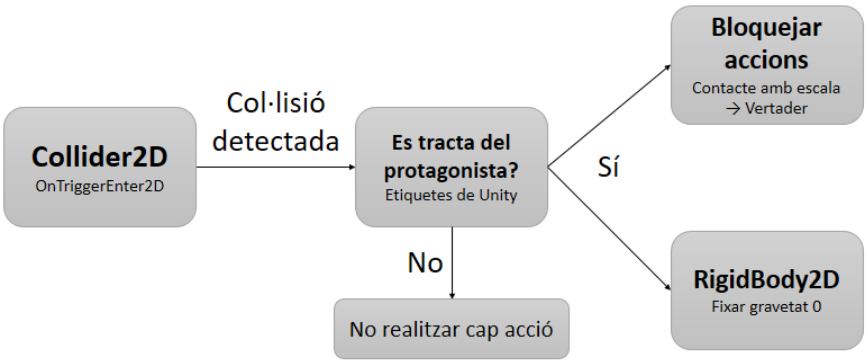

Figura 7.11. Entrada en la col·lisió

En primer lloc, mitjançant la funció *OnTriggerEnter2D* que s'executa just en el moment en que es detecta el contacte, es defineix el booleà de contacte com a vertader, que bloqueja

tots els moviments del personatge excepte córrer i saltar. També es canvia la gravetat del personatge, imposant un valor de 0.

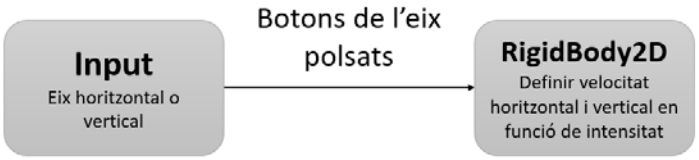

Figura 7.12. Accions mentre es manté el contacte

Mentre el contacte es manté, a cada fotograma s'executa la funció *OnTriggerStay2D*, que fixa el valor de la velocitat del personatge en funció dels eixos verticals i horitzontals, permetent així el moviment vertical al llarg de l'escala i la possibilitat de sortir de la col·lisió amb l'escala.

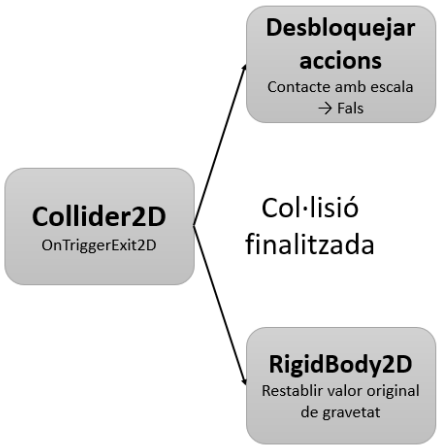

Figura 7.13. Accions realitzades en finalitzar la col·lisió

Finalment, quan es surt de la col·lisió, s'executa la funció *OnTriggerExit2D*, que defineix el booleà de contacte com a fals i restableix la gravetat del personatge, permetent que es calculin les físiques amb normalitat.

7.3. Nivells

Únicament s'ha implementat un nivell de forma complerta, que és el Laboratori. També s'ha creat el nivell del poble però les funcionalitats especificades a la fase de disseny no s'han implementat. Com s'ha mencionat anteriorment, el nivell s'ha creat al software *Tiled* i posteriorment, utilitzant la utilitat *Tiled2Unity* s'ha importat a *Unity*.

En el disseny del nivell s'han incorporat cartells a mode de tutorial per mostrar al jugador

quines eines té a la seva disposició per superar diferents situacions dins del joc, com es pot veure a la **Figura 7.14**.

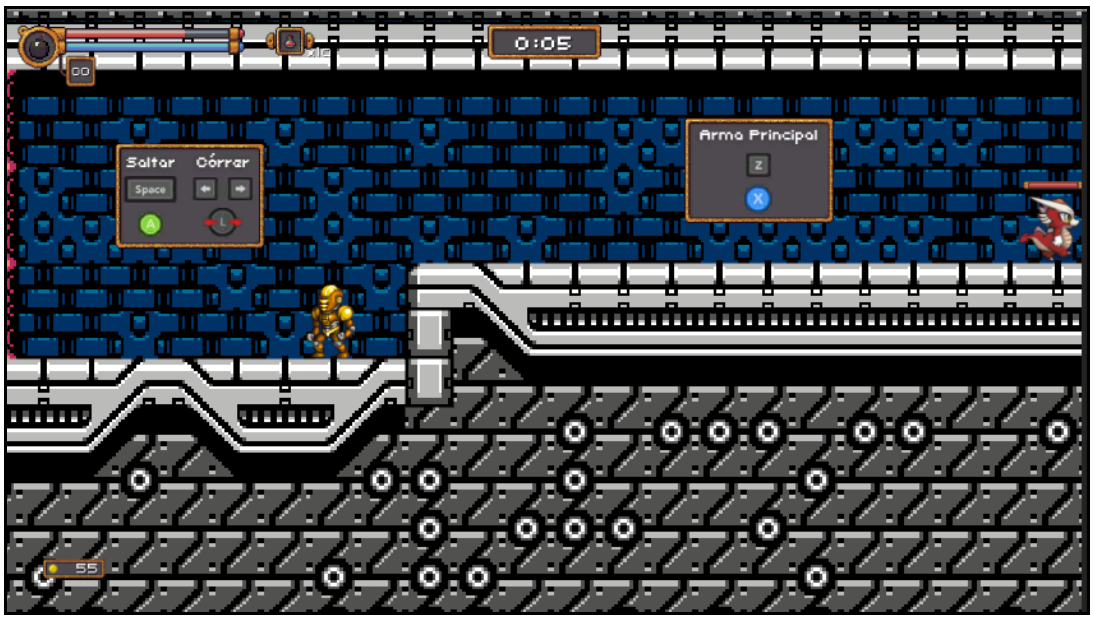

Figura 7.14. Captura del nivell 0, Laboratori

Per aquest nivell també s'han implementat dues sales secretes, que només poden ser descobertes si es llança una bomba contra la paret. Per implementar aquesta característica s'ha creat un *GameObject* de paret i s'ha superposat al terreny real del nivell, de forma que el jugador no el pugui veure. Addicionalment, s'ha modificat el comportament de la bomba per tal que a més de causar danys als enemics destrueixi els objectes d'aquest tipus, creant així l'efecte desitjat. A la ***Figura 7.15*** es pot veure un exemple.

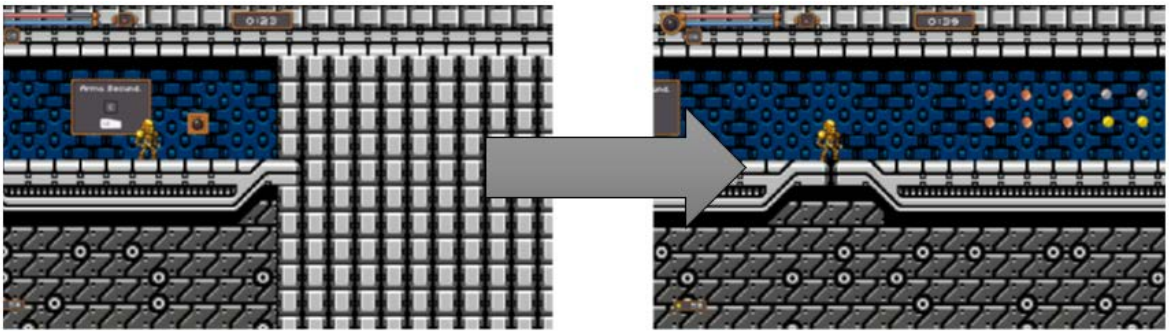

Figura 7.15. Sala secreta

A continuació es detalla la llista d'enemics i objectes utilitzats al nivell.

Nom	Tipus	Quantitat
Drac	Enemic	2
Torreta	Enemic	3
Mini torreta	Enemic	4
Moneda de coure	Objecte	17
Moneda de plata	Objecte	4
Moneda d'or	Objecte	4
Munició de bomba	Objecte	5
Munició de làser	Objecte	6
Escales	Objecte	6

8. Interfície d'usuari, controls i sons

Un cop implementades tots els elements del joc, cal crear menús per a que el jugador pugui navegar entre les escenes que conformen el joc, una interfície d'usuari dins del joc que aporti la informació necessària durant el transcurs de la partida i també definir quins són els controls utilitzats en el videojoc. Addicionalment, s'afegeixen efectes sonors i música de fons per millorar l'experiència de joc.

8.1. Interfície d'usuari i menús

La interfície d'usuari s'ha desenvolupat utilitzant les eines que incorpora l'editor de *Unity*, així com la classe *UI*. El funcionament és molt intuïtiu, doncs els components que Unity incorpora per aquesta tasca estan preparats per implementar qualsevol tipus de menú. En els menús del prototip s'ha treballat bàsicament amb els components *Text*, que com el seu nom indica permet mostrar text per pantalla i es fàcilment modificable des dels *scripts*, i també amb el component *Button,* que crea un botó al que únicament cal associar una funció que realitzi les accions requerides. A continuació es mostren els menús que s'inclouen al prototip, que generalment compleixen les especificacions del disseny (pàgina 18).

Figura 8.1. Menú principal

Figura 8.2. Introducció a la trama

Figura 8.3. Mapa del món

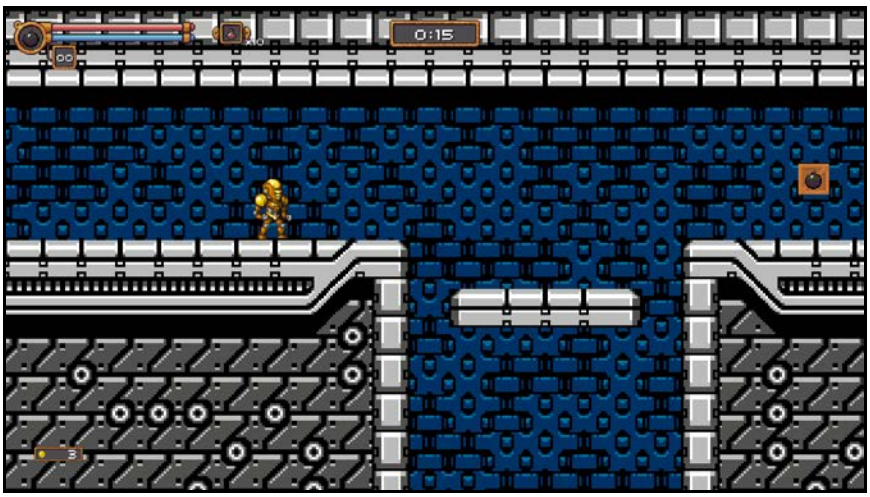

Figura 8.4. Interfície dins del joc

Figura 8.5. Pantalla de fi de nivell

Figura 8.6. Menú de pausa

Figura 8.7. Crèdits

8.2. Controls

Com s'ha mencionat anteriorment, els controls es defineixen mitjançant el *InputManager* incorporat a *Unity*, assignant els botons dels dispositius d'entrada que realitzaran les funcions disponibles. En el cas del projecte, degut a que es dissenya un joc per a computador pels sistemes Windows, Mac Os i Linux, s'han pres com a dispositius d'entrada el teclat i els comandaments de joc, prenent com a referent el comandament de Xbox 360 de Microsoft, degut a que són bastant utilitzats en jocs per computador i ja es disposava de comandaments d'aquest tipus abans de començar el projecte. A continuació es mostra una taula amb la distribució de controls, en referència als esquemes dels dispositius d'entrada representats a la **Figura 8.8**.

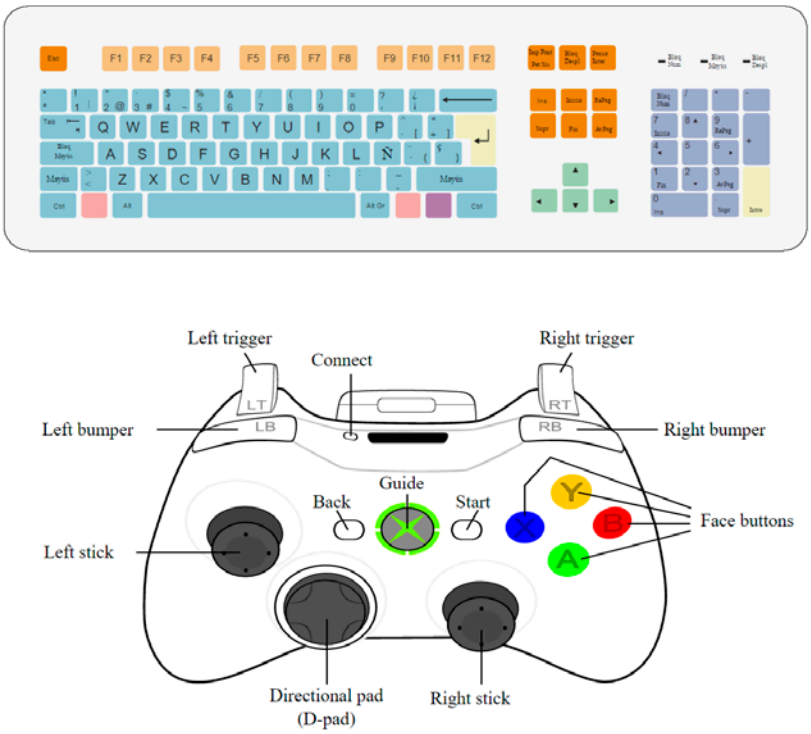

Figura 8.8. Distribució de botons dels dispositius d'entrada

Acció	Descripció	Teclat	Comand.
Eix horitzontal	Controlar desplaçament del protagonista, navegar entre menús		
Eix Vertical	Apuntar amunt, desplaçar-se al llarg d'escales, navegar entre menús		

Arma principal	Utilitzar l'arma principal		
Arma secundària	Utilitzar l'arma secundària		
Canviar arma secundària	Canvia a la següent arma secundària		
Saltar	En les pantalles de joc, el personatge salta		
Rodar	En les pantalles dins del joc, el personatge roda		
Objectes consumibles	Utilitzar l'objecte consumible equipat (en el cas del prototip únicament es disposa de pocions)		
Pausa	En les pantalles dins del joc, obre el menú de pausa		
Confirmar	En les pantalles fora del joc, realitza les accions del menú seleccionat		
Cancel·lar	En les pantalles fora de joc, retorna al menú anterior sense realitzar cap acció		

8.3. Sons

Pel que fa als sons, al prototip únicament s'ha incorporat música de fons. Per implementar la música de fons, només cal afegir el component *Audio Source* al *GameObject* que contingui la càmera utilitzada per renderitzar el joc, que permet adjuntar un fitxer d'audio. D'aquesta manera, degut a que únicament es té una càmera per escena, la música de fons es reprodueix en tot moment. Totes les escenes del prototip contenen música de fons.

Com en el cas dels recursos artístics, crear la música necessària per cada nivell requereix d'una gran quantitat de temps i coneixement, per tant en l'elaboració del treball s'ha optat per incorporar música d'ús lliure.

9. Característiques no incloses al prototip

Degut a les limitacions en quant a temps imposades per les característiques del projecte, el prototip no inclou totes les especificacions definides a la fase de disseny, que haurien de ser realitzades de forma posterior per tal de completar el desenvolupament del videojoc. El llistat de característiques que no han estat incloses és el que es mostra a continuació, en referència a l'apartat de disseny (pàgines 1010 a 30).

- **Mecàniques de joc:** Punts de control, millora progressiva del personatge, diferents finals.
- **Personatge principal:**
    - Moviments: Ajupir-se.
    - Armes: Canó làser mode foc carregat, cops físics.
    - Habilitats: Doble salt, escut.
- **Personatges aliats / neutrals:** Toke, Waldan, botiguer.
- **Enemics:** Enemics finals, patrulla amb dispar, patrulla amb explosió.
- **Objectes col·leccionables:** Millores, apunts.
- **Objectes consumibles:** Beguda energètica, bateria.
- **Objectes interactius:** Cartells, portes.
- **Interfície gràfica d'usuari:** Opcions, extres.
- **Nivells:** Nivells 1, 2, 3, 4, 5 i 6.

10. Planificació i pressupost

10.1. Planificació

En aquest apartat es detalla la planificació de les diferents etapes requerides en el projecte. La planificació ha estat realitzada tenint en compte els 4 mesos disponibles per la realització del primer prototip (Setembre 2015 – Gener 2016), en cap cas es correspon al projecte de videojoc totalment acabat.

Les fases del projecte són:

- **Recerca d'informació (0,5 setmanes):** Etapa prèvia al començament del treball de fi de grau. Durant aquesta etapa es va buscar informació sobre els motor de joc 2D disponibles de forma gratuïta, com per exemple quin llenguatge utilitzen, les funcionalitats que incorporen, etc.

- **Aprenentatge de *C#* (1 setmana):** Aquí comença la realització del treball, un cop decidit que el motor a utilitzar seria *Unity* i el llenguatge *C#*, es va començar l'aprenentatge del llenguatge, doncs els coneixements previs eren únicament *Python*.

- **Aprenentatge de l'eina *Unity* (2,5 setmanes):** Havent adquirit uns coneixements bàsics de *C#*, es va començar a treballar amb *Unity*, investigant en la documentació i els tutorials disponibles a la pròpia web de *Unity*.

- **Disseny conceptual del videojoc (2 setmanes):** De manera gairebé paral·lela a l'aprenentatge de *Unity*, es va realitzar el disseny conceptual del videojoc, documentat a l'apartat 4 d'aquesta memòria. En aquesta fase es va definir com seria el joc final, i quines mecàniques de joc i elements cal implementar en *Unity*.

- **Desenvolupament del prototip (10 setmanes):** Un cop es va realitzar el disseny conceptual del videojoc i es van adquirir uns coneixements de *Unity* acceptables, es va realitzar el desenvolupament del prototip. Es tracta de la part que ha consumit més temps dins del projecte.

- **Documentació de la memòria (3 setmanes):** Entrant en la part final del desenvolupament del prototip, s'ha procedit amb el redactat de la memòria.

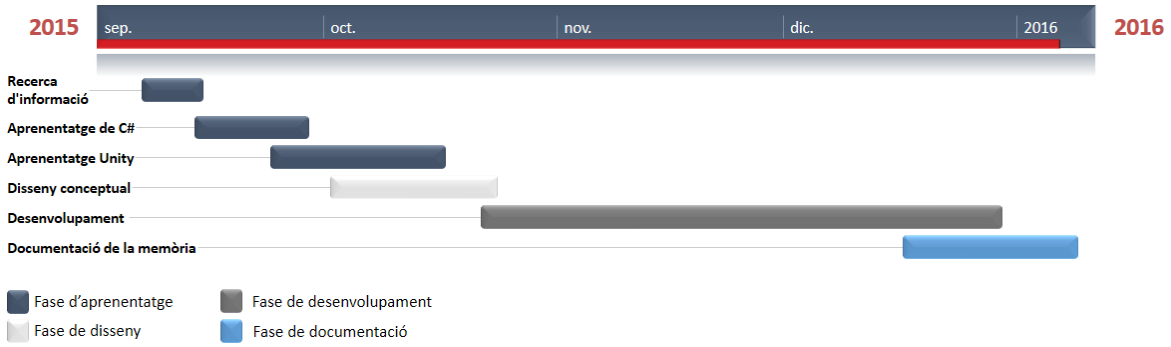

Figura 10.1. Diagrama de Gantt amb les fases del projecte

10.2. Pressupost

En l'apartat de pressupost es detallen els costos per la realització del projecte, que consisteixen en hores de treball dels enginyers participants i despeses en hardware o software. Es considera que la recerca d'informació per posar en marxa el projecte no està remunerada.

Pel que fa a les hores, es calcula que de mitja han estat unes 20 hores de dedicació per setmana. Es considera que el sou del graduat en tecnologies industrials és de 25 € / hora per les fases de desenvolupament i redactat de la memòria i 18 € / hora en les fases d'aprenentatge i disseny.

El software i recursos utilitzats són gratuïts, per tant no han suposat cap cost. Tampoc es considera els costos del hardware utilitzat, ja que no ha estat adquirit per la realització del projecte si no que ja es disposava d'ell.

Concepte	Treballador	Dedicació (h)	Cost (€/h)	Cost fase (€)
Aprenentatge C#	Graduat Enginyeria Industrial	20	18	360
Aprenentatge Unity	Graduat Enginyeria Industrial	50	18	900
Disseny conceptual	Graduat Enginyeria Industrial	40	18	720
Desenvolupament	Graduat Enginyeria Industrial	200	25	5000
Redactat memòria	Graduat Enginyeria Industrial	60	25	1500
Total		**370**		**8480**

Figura 10.2. Taula de costos per la creació del prototip

La ***Figura 10.2*** mostra les despeses del projecte, que té un cost total de **8480 €**. Com a l'apartat de planificació, aquesta quantitat es correspon a la quantitat necessària per realitzar el prototip, per a calcular els costos per obtenir el joc final caldria detallar les fases del projecte sencer. També cal afegir que en cas de comercialitzar el joc, caldria obtenir una llicència de la versió professional de *Unity*, que té un cost de 75 $ / mes, aproximadament uns 70 € segons el canvi de moneda actual.

11. Consideracions mediambientals

Degut a les característiques del projecte, que té com a objectiu el desenvolupament d'un videojoc, no es produeixen accions perjudicials contra el medi ambient. Si bé es cert que per desenvolupar el joc calen ordinadors, que consumeixen electricitat i s'escalfen, es considera que l'impacte ambiental que poden causar és negligible, ja que no existeix una diferència apreciable entre la energia utilitzada al projecte i el consum elèctric d'un habitatge mitjà, per exemple.

Conclusions

S'han realitzat de manera exitosa els objectius plantejats a l'inici del projecte, que eren els de dissenyar un videojoc i desenvolupar un prototip que incorporés les seves característiques principals. La realització d'aquest projecte ha estat una gran experiència, doncs he adquirit molts coneixements sobre el desenvolupament de videojocs, així com el llenguatge C#, que no havia utilitzat amb anterioritat. Addicionalment, treballar amb videojocs 2D també m'ha aportat molt en quant al ús de mapes basats en *tiles* i personatges basats en *sprites*. Tractar la part de disseny conceptual també ha estat molt interessant, ja que tot i no semblar-ho quan es consumeix el producte, realitzar un bon disseny és essencial per a crear un bon videojoc.

Pel que fa a l'eina utilitzada en la majoria d'aspectes del , *Unity*, crec que és una eina molt versàtil, que permet desenvolupar qualsevol tipus de joc, ja sigui en 2D i 3D, tal com es pot veure en el mercat, ja que en els últims anys el nombre de jocs comercials desenvolupats a *Unity* ha crescut de manera important. El tractament dels elements de joc com a objectes i la incorporació de una gran varietat de components permet que des dels usuaris més novells fins als més experts puguin trobar la forma d'implementar les seves idees sense grans dificultats. Un altre punt positiu sobre *Unity*, és la gran comunitat de que disposa, doncs quan he tingut dubtes sobre com implementar alguns aspectes concrets del joc, la majoria del temps he pogut solucionar-los gràcies als fòrums de la comunitat.

En definitiva, la realització d'aquest projecte ha estat molt satisfactòria i, tot i que personalment m'hauria agradat crear un prototip molt més complert i més semblant a la idea que tenia en ment en iniciar el treball, crec que l'esforç s'ha vist recompensat per la gran quantitat de coneixements sobre el tema que he adquirit.

Agraïments

En primer lloc m'agradaria agrair al director del treball, Lluis Solano, per l'ajuda i bona disposició mostrada al llarg del treball, així com la seva assistència pel que fa al redactat de la memòria.

També m'agradaria agrair a tots els artistes que decideixen compartir les seves obres amb la comunitat de manera desinteressada, sense ells aquest projecte no hauria estat possible.

Bibliografia

Referències bibliogràfiques

[1] AEVI, Asociación Española de Videojuegos. A14, anuario de la industria del videojuego. Madrid: 2015, p. 62-72

Bibliografia complementària

[1] LIBERTY, J., MACDONALD, B. *Learning C# 3.0*. Sebastopol, CA: O'Reilly Media. Novembre 2008

[2] SCOLASTICI, C., NOLTE, D. *Mobile Game Design Essentials.* Birmingham, UK: Packt Publishing. Novembre 2013

[3] BETHKE, E. *Game development and production*. Plano, TX: Wordware Publishing Inc. 2003.

[4] CALABRESE, D. *Unity 2D Game Development. Birmingham*, UK: Packt Publishing. Març 2014.

[5] UNITY. Unity – Learn.
[http://unity3d.com/es/learn, Setembre 2015 – Gener 2016]

[6] UNITY. Unity – Scripting Api.
[http://docs.unity3d.com/ScriptReference/, Setembre 2015 – Gener 2016]

[7] UNITY. Unity Community.
[http://forum.unity3d.com/, Setembre 2015 – Gener 2016]

[8] OPENGAMEART. OpenGameArt.org
[http://opengameart.org/, Setembre 2015 – Gener 2016]

[9] MACHINIMASOUND. Royalty Free Music for Games & Films.
[https://machinimasound.com/, 12 de Gener de 2016]